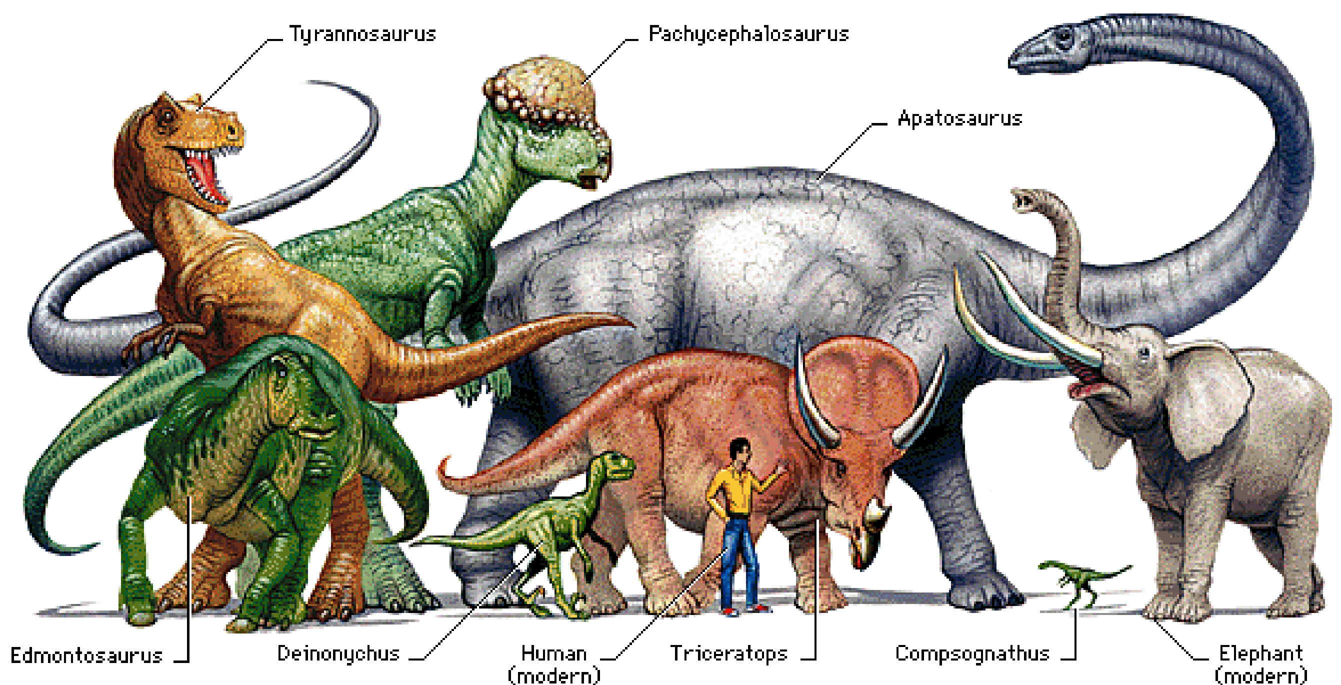
พื้นฐานระบบปฏิบัติการยูนิกซ์
(Fundamentals of UNIX)



ผศ.สุชาติ คุ้มมะณี

พิมพ์เผยแพร่ 15 ตุลาคม 2558 แก้ไขล่าสุด 15 ตุลาคม 2558

พื้นฐานระบบปฏิบัติการยูนิกซ์
(Fundamentals of UNIX)



Asst. Prof. Suchart Khummanee
Email: suchart.k@msu.ac.th

คำนำ

วิชาพื้นฐานระบบปฏิบัติการยูนิกซ์ เป็นการแนะนำระบบยูนิกซ์ การเข้าใช้งานระบบยูนิกซ์ การใช้คำสั่งพื้นฐานต่างๆ ในการจัดการเกี่ยวกับระบบไฟล์ ยูทิลิตี้ในระบบยูนิกซ์ ยูทิลิตี้สำหรับการประมวลผลไฟล์ การควบคุมโปรเซส การบีบอัดข้อมูล การสำรองข้อมูล การติดตั้งและประยุกต์ใช้ซอฟต์แวร์ ระบบรักษาความปลอดภัยของผู้ใช้ยูนิกซ์ ระบบการสื่อสาร ระหว่างผู้ใช้และ ระบบรายการสื่อสารแบบทางไกล การแก้ไขไฟล์ด้วยเอดิเตอร์ การเขียนโปรแกรมบนยูนิกซ์โปรแกรมประยุกต์บนยูนิกซ์

เนื่องจากวิชาดังกล่าวจำเป็นต้องมีการปฏิบัติควบคู่ไปกับภาคทฤษฎีเพื่อเสริมทักษะให้นิสิต/นักศึกษา เกิดความเข้าใจ มีความชำนาญ และประยุกต์เอาความรู้ที่ได้จากภาคทฤษฎีมาบูรณาการร่วมกับการปฏิบัติได้ ดังนั้นในเอกสารเล่มนี้จะมีตัวอย่างและ โจทย์ให้นิสิต/นักศึกษาได้ลงมือปฏิบัติค่อนข้างมาก

หากเอกสารดังกล่าวมีข้อปรกพร่องอันใดผู้เขียนต้องขออภัยไว้ ณ ที่นี้ด้วย

ผศ. สุชาติ คุ้มมะณี

Suchart.k@msu.ac.th

สารบัญ

หน้า



บทที่

1

ประวัติยูนิกซ์และลินุกซ์

วัตถุประสงค์

1. สามารถอธิบายประวัติศาสตร์และวิวัฒนาการของระบบปฏิบัติการยูนิกซ์และลินุกซ์ได้
2. สามารถอธิบายบุคคลผู้ให้กำเนิดระบบปฏิบัติการยูนิกซ์และลินุกซ์ได้
3. สามารถอธิบายวิวัฒนาการของยูนิกซ์และลินุกซ์สายพันธุ์ต่างๆ (distribution) ได้

บทที่ 1
ประวัติยูนิกซ์และลินุกซ์

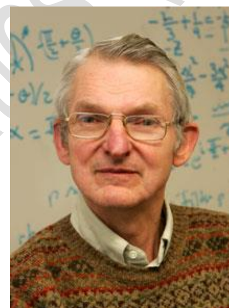
บทนำ

ในบทนี้จะอธิบายถึงจุดกำเนิดและวิวัฒนาการของระบบปฏิบัติการยูนิกซ์และลินุกซ์ ตั้งแต่อดีตจนถึงปัจจุบัน ผู้ที่มีบทบาทสำคัญในการสร้างระบบปฏิบัติการ สถานที่เกิดเหตุการณ์ต่างๆ คุณสมบัติที่โดดเด่นของระบบปฏิบัติการ และสายพันธ์ของยูนิกส์และยูนิกส์ที่เกิดขึ้นในปัจจุบัน

ยูนิกซ์(UNIX) และลินุกซ์(Linux) เป็นระบบปฏิบัติการคอมพิวเตอร์ที่มีความสามารถทำงานได้พร้อมๆ กันหลายๆ งาน(Multitasking) และมีผู้ใช้งานในเวลาเดียวกันได้หลายคน (Multiuser) เริ่มพัฒนาโดยกลุ่มพนักงานในห้องปฏิบัติการ AT&T Bell Labs โดยกลุ่มนักพัฒนาที่เป็นที่รู้จัก คือ Ken Thompson, Dennis Ritchie และ Douglas McIlroy [1]


Ken Thompson


Dennis Ritchie


Douglas McIlroy

เหตุการณ์ที่เกิดขึ้นในประวัติศาสตร์ของยูนิกซ์หรือลินุกซ์ [2]

ประวัติยูนิกซ์

ปี ค.ศ. 1957 Bell Labs ต้องการสร้างระบบปฏิบัติการที่มีความสามารถรองรับการทำงาน (Batch jobs) ที่แตกต่างกันได้หลายๆ งาน จึงได้สร้างระบบปฏิบัติการครั้งแรกชื่อว่า BESYS

ปี ค.ศ. 1965 สถาบันเทคโนโลยีแมสซาชูเซตส์ (MIT) , AT&T Bell Labs และบริษัท General Electric ได้ร่วมมือกันวิจัยระบบปฏิบัติการที่ชื่อว่า Multics (ย่อมาจาก Multiplexed Information and Computing Service) โดยมีจุดมุ่งหมายเพื่อทำงานบนเครื่องเมนเฟรมรุ่น GE-645 แต่ภายหลัง AT&T ได้ถอนตัวออกจากโครงการนี้

ปี ค.ศ. 1969 Ken Thompson ซึ่งเป็นหนึ่งในทีมพัฒนาในขณะนั้น ได้เขียนเกมบนเครื่อง GE-645 ชื่อว่าเกม Space Travel และพบปัญหาว่าเกมทำงานได้ช้าและเสียค่าใช้จ่ายมากกว่าที่ควร เขาจึงย้ายเกมส์มาทำงานใหม่บนเครื่อง PDP-7 ของบริษัท DEC แทน ด้วยภาษาแอสเซมบลี โดยความช่วยเหลือของ Dennis Ritchie ประสบการณ์เหล่านี้ทำให้ Thompson หันมาพัฒนา

ระบบปฏิบัติการบนเครื่อง PDP-7 ระบบปฏิบัติการนี้มีชื่อว่า UNICS ย่อมาจาก Uniplexed Information and Computing System เนื่องจากว่าการออกเสียงสามารถสะกดได้หลายแบบ และพบปัญหาชื่อใกล้เคียงกับ Multics ภายหลังจึงเปลี่ยนชื่อเป็น UNIX ในปีนี้มีบุคคลสำคัญได้ถือกำเนิดขึ้นมาคือ Linun Torvalds(ผู้ให้กำเนิด ลินุกซ์ ในอนาคต)

ปี ค.ศ. 1970 การพัฒนายูนิกซ์ในช่วงนี้ยังไม่ได้รับความสนับสนุนด้านการเงินจาก Bell Labs เมื่อระบบพัฒนามากขึ้น Thompson และ Ritchie จึงสัญญาว่าจะเพิ่มความสามารถในการประมวลผลคำ (Word Processing) บนเครื่อง PDP-11/20 และเริ่มได้รับการตอบรับจาก Bell Labs ในปีค.ศ. 1970 ระบบปฏิบัติการยูนิกซ์จึงได้รับการเรียกชื่ออย่างเป็นทางการ

ปี ค.ศ. 1971 กำเนิดหนังสือยูนิกซ์เล่มแรกชื่อ UNIX Programmer's Manual ตีพิมพ์ครั้งแรกวันที่ 3 พฤศจิกายน ค.ศ. 1971 ผู้แต่งคือ K. Thompson และ D. M. Ritchie มีคำสั่งให้ใช้งานทั้งหมด 60 คำสั่ง เช่น b (compile B program); boot (reboot system); cat (concatenate files); chdir (change working directory); chmod (change access mode); chown (change owner เป็นต้น

ปี ค.ศ. 1972 หนังสือยูนิกซ์ทำการตีพิมพ์เป็นครั้งที่สองวันที่ 12/06/1972 และ Ritchie ได้เริ่มเขียนระบบปฏิบัติการขึ้นมาใหม่ด้วยภาษาซี ทำให้สะดวกต่อการนำยูนิกซ์ไปทำงานบนเครื่องชนิดอื่นมากขึ้น ทาง AT&T ได้เผยแพร่ยูนิกซ์ไปยังมหาวิทยาลัย และหน่วยงานต่างๆ ของรัฐบาลโดยสัญญาการใช้งานเปิดเผยซอร์สโค้ด ยกเว้นเคอร์เนลส่วนที่ยังเขียนด้วยภาษาแอสเซมบลี

ปี ค.ศ. 1973 หนังสือยูนิกซ์ทำการตีพิมพ์เป็นครั้งที่ 3 และ 4 เมื่อ กุมภาพันธ์ 1973, พฤศจิการยน 1973 ตามลำดับ

ปี ค.ศ. 1974 หนังสือยูนิกซ์ทำการตีพิมพ์เป็นครั้งที่ 5 เมื่อ มิถุนายน 1974

ปี ค.ศ. 1975 ยูนิกซ์เวอร์ชัน 4, 5 และ 6 ออกมาใช้งานในปี ค.ศ. 1975 ได้เพิ่มคุณสมบัติ pipe เข้ามา ยูนิกซ์เวอร์ชัน 7 ซึ่งเป็นเวอร์ชันสุดท้ายที่พัฒนาแบบการวิจัย

ปี ค.ศ. 1979 ยูนิกซ์เวอร์ชัน 8, 9 และ 10 ออกมาในภายหลังในทศวรรษที่ 80 ในวงจำกัดเฉพาะมหาวิทยาลัยบางแห่งเท่านั้น

ปี ค.ศ. 1982 AT&T นำยูนิกซ์ 7 มาพัฒนาและออกขายในชื่อ Unix System III แต่บริษัทลูกของ AT&T ชื่อว่า Western Electric ยังคงนำยูนิกซ์รุ่นเก่ามาขายอยู่เช่นกัน เพื่อยุติความสับสนทางด้านชื่อ AT&T จึงรวมการพัฒนาทั้งหมดจากบริษัทและมหาวิทยาลัยต่างๆแล้วตั้งชื่อว่า Unix System V ซึ่งได้รวมเอาโปรแกรม vi ที่พัฒนาโดย Berkeley Software Distribution (BSD) จากมหาวิทยาลัยแคลิฟอร์เนีย เบิร์กลีย์ รวมอยู่ด้วย ยูนิกซ์รุ่นนี้สามารถทำงานได้บนเครื่อง VAX ของบริษัท DEC ยูนิกซ์ในขณะนั้นจะเป็นแบบเชิงการค้า ผู้ใช้จำเป็นต้องเสียเงินในการซื้อระบบปฏิบัติการและไม่เปิดเผยซอร์สโค้ดด้วย ทางมหาวิทยาลัยแคลิฟอร์เนีย เบิร์กลีย์ จึงพัฒนายูนิกซ์ของตัวเองแบบแจกให้สามารถใช้งานได้ฟรี ซึ่งเป็นทางเลือกสำหรับผู้ที่ใช้งาน System V อยู่แล้ว การพัฒนาที่สำคัญที่สุดคือเพิ่มการสนับสนุนโพรโทคอลสำหรับเครือข่าย TCP/IP เข้ามา

ในขณะที่บริษัทอื่นๆ เริ่มพัฒนายูนิกซ์บนเครื่องคอมพิวเตอร์ของตนเอง โดยส่วนมากใช้ยูนิกซ์ที่ซื้อสัญญามาจาก System V แต่บางบริษัทเลือกพัฒนาจาก BSD แทน หนึ่งในทีมพัฒนาของ BSD คือ Bill Joy มีส่วนในการสร้าง SunOS (ปัจจุบันคือ โซลาริส) ของบริษัทซัน ไมโครซิสเต็มส์

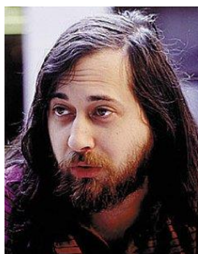
- ทีมพัฒนา BSD ได้ออกจากมหาวิทยาลัยและก่อตั้งบริษัท Berkeley Software Design, Inc (BSDI) เป็นบริษัทแรกที่นำ BSD มาขายในเชิงการค้า ในภายหลังเป็นต้นกำเนิดของระบบปฏิบัติการ FreeBSD, OpenBSD และ NetBSD, AT&T ยังคงพัฒนาความสามารถต่างๆ เข้าสู่ยูนิกซ์ System V และรวมเอา Xenix (ยูนิกซ์ของบริษัทไมโครซอฟท์) , BSD และ SunOS เข้ามารวมใน System V Release 4 (SVR4) เพื่อเป็นผลิตภัณฑ์หนึ่งเดียวสำหรับลูกค้า ซึ่งเพิ่มราคาขึ้นอีกมาก

หลังจากนั้นไม่นาน AT&T ขายสิทธิ์ในการถือครองยูนิกซ์ให้กับบริษัทโนเวลล์ และโนเวลเองได้สร้างยูนิกซ์ของตัวเองที่ชื่อ UnixWare ซึ่งพัฒนามาจากระบบปฏิบัติการ NetWare เพื่อแข่งกับระบบปฏิบัติการวินโดวส์เอ็นทีของไมโครซอฟท์

ปี ค.ศ. 1995 บริษัทโนเวลล์ขายส่วนต่างๆ ของยูนิกซ์ให้กับบริษัท Santa Cruz Operation (SCO) โดยโนเวลล์ยังถือลิขสิทธิ์ของยูนิกซ์ไว้ ค.ศ. 2000 SCO ขายสิทธิ์ส่วนของตนเองให้กับบริษัท Caldera ซึ่งเปลี่ยนชื่อภายหลังเป็น SCO Group [3]

ประวัติลินุกซ์ [4]

ปี ค.ศ. 1983 ริชาร์ด สตอลแมน (Richard Stallman) ได้ก่อตั้งโครงการกนูขึ้น มีจุดมุ่งหมายคือ ต้องการพัฒนาระบบปฏิบัติการคล้ายยูนิกซ์ที่เป็นซอฟต์แวร์เสรีทั้งระบบ ราวช่วง ค.ศ. 1990 โครงการกนูมีส่วนโปรแกรมที่จำเป็นสำหรับระบบปฏิบัติการเกือบครบทั้งหมด ได้แก่ คลังโปรแกรม (Libraries) คอมไพเลอร์ (Compiler) โปรแกรมแก้ไขข้อความ(Text Editor) และเปลือกระบบยูนิกซ์(Shell) ซึ่งขาดแต่เพียงเคอร์เนล(Kernel) เท่านั้น ในปลายปี ค.ศ. 1990 โครงการกนูได้พัฒนาเคอร์เนลชื่อ Hurd เพื่อใช้ในระบบกนูซึ่งในขณะนั้นมีปัญหาเกี่ยวกับความเร็วในการประมวลผล



Richard Stallman

ปี ค.ศ. 1987 ศาสตราจารย์ Andrew S.Tanenbaum ได้ออกแบบสร้าง ยูนิกซ์สำหรับเครื่องไมโครคอมพิวเตอร์ ซึ่งสามารถทำงานได้ทั้งบนเครื่อง PC, Mac, Amiga โดยให้ชื่อว่า Minix และยังแจกซอร์สโค้ดฟรีให้แก่นักวิจัยเพื่อนำไปพัฒนาต่อ



Profressor Andrew S.Tanenbaum

ปี ค.ศ. 1991 ผู้เริ่มพัฒนาลินุกซ์ เคอร์เนลเป็นคนแรก คือ ลินุส โตร์วัลดส์ (Linus Torvalds) ชาวฟินแลนด์ เมื่อสมัยที่เขายังเป็นนักศึกษาคอมพิวเตอร์ ที่มหาวิทยาลัยเฮลซิงกิ โตร์วัลดส์เริ่มโครงการพัฒนาเคอร์เนล ขณะศึกษาในมหาวิทยาลัยแล้ว โดยอาศัย Minix ซึ่งเขียนขึ้นโดยศาสตราจารย์ Andrew S.Tanenbaum เป็นระบบที่คล้ายกับ Unix ซึ่งมากับหนังสือเรื่องการออกแบบระบบปฏิบัติการของศาสตราจารย์ Andrew S.Tanenbaum นั่นเอง มาเป็นเป็นต้นแบบในการเขียนขึ้นใหม่ Torvalds ได้พัฒนาโดยใช้ IA-32 assembler และภาษาซี คอมไพล์เป็นไฟล์ไบนารี่และบูทจากแผ่นฟลอปปี้ดิสก์ เขาได้พัฒนามาเรื่อยๆจนกระทั่งสามารถบูทตัวเองได้ (กล่าวคือสามารถคอมไพล์ภายในลินุกซ์ได้เลย)



Linus Torvalds

การพัฒนาในช่วงแรกมีการแจกจ่ายซอร์สโค้ดของลินุกซ์บนอินเตอร์เน็ต ผลจาการแจกโค้ด ทำให้เกิดการร่วมมือกันพัฒนาอย่างแพร่หลายทั่วโลกลินุกซ์ เจิรญเติบโตอย่างรวดเร็วและเพียบพร้อมไปด้วยฟังชันที่มีความหลากหลายคล้าย UNIX ในช่วงต้นในการพัฒนา ลินุกซ์ จะเน้นที่ kernel ของระบบปฏิบัติการ(kernel ทำหน้าที่หลักในการจัดการทรัพยากรของระบบทั้งหมด เช่น โปรเซส (Process), การจัดเวลาซีพียู (CPU Scheduling), การจัดการหน่วยความจำ (Memory Management), การจัดการไฟล์ (File Management), การจัดการอุปกรณ์อินพุต / เอาต์พุต (Input/Output Devices Management) เป็นต้น

Linux Kernel เวอร์ชัน 0.01 ออกเผยแพร่ ในเวอร์ชันนี้ยังไม่มีความสามารถมารถทางด้านเน็ตเวิร์ค ทำงานได้เฉพาะโปรเซสเซอร์ที่ทำงานร่วมกัน(Compatible) ได้กับหน่วยประมวลผลกลางอินเทล 80386 มีข้อจำกัดในการสนับสนุนดีไวด์ไดร์เวอร์(Device Driver) รองรับการทำงานของหน่วยความจำเสมือน โครงสร้างของระบบไฟล์ยังใช้แบบ Minix เดิมอยู่

ปี ค.ศ. 1994 ลินุกซ์ เวอร์ชัน 1.0 ออกเผยแพร่ใช้เวลาในการพัฒนาถึง 3 ปี มีการเพิ่มฟังก์ชันและคุณสมบัติอื่นๆ เข้าไปมากมาย แต่มีคุณสมบัติที่สำคัญเป็นอย่างยิ่งคือ ความสามารถ

ทางด้านเน็ตเวิร์ค โดยในเวอร์ชันนี้สนับสนุน โพรโทคอล TCP/IP ซึ่งเป็นมาตรฐานหลักของ UNIX และสามารถงานร่วมกับซ็อกเก็ตอินเทอร์เฟซของ BSD ในการเขียนโปรแกรมทางด้านเน็ตเวิร์คได้เป็นอย่างดี สนับสนุนดีไวซ์ไดร์เวอร์ที่สามารถทำงานร่วมกับอีเทอร์เน็ตโทโพโลยี (Ethernet Topology) สนับสนุนระบบไฟล์ให้มีความสามารถเพิ่มสูงขึ้น จากเดิมที่ทำงานได้เฉพาะกับระบบไฟล์ของ Minix สนับสนุนคอนโทรลเลอร์แบบ SCSI สำหรับการเข้าถึงข้อมูลในดิสก์ ขยายขีดความสามารถในการอ้างอิงหน่วยความจำแบบเสมือนแบบ page file กับ swap file สนับสนุนฮาร์ดแวร์มากขึ้น แต่ยังคงมีขีดจำกัดเฉพาะโปรเซสเซอร์อินเทลอยู่ นอกจากนี้ยังสนับสนุนด้านฮาร์ดแวร์ที่ต่อพ่วงภายนอกเพิ่มขึ้น เช่น ฟล็อปปี้ดิสก์ ซีดีรอม การ์ดเสียง เมาส์และคีบอร์ดของต่างประเทศ มีการจำลองผลการคำนวณทางด้านคณิตศาสตร์ใน 80386 สำหรับผู้ใช้ที่ไม่มีแมตโครโปรเซสเซอร์ที่มีอยู่ใน 80387 สนับสนุนการติดต่อระหว่างโปรเซส IPC ในรูปแบบของ System V บน UNIX รวมทั้งการแชร์หน่วยความจำ, semaphore และแมสเสจคิว

ปี ค.ศ. 1995 ลินุกซ์ เวอร์ชัน 1.2 ออกเผยแพร่ เพิ่มเติมในการสนับสนุนฮาร์ดแวร์ที่หลากหลายขึ้น รวมถึงสถาปัตยกรรมบัสแบบ PCI สนับสนุนโพรโทคอล IPX และเพิ่มฟังก์ชันในการกำหนดไฟล์วอลล์ และสามารถรองรับซีพียู SPARC, Alpha และ MIPS ได้

ปี ค.ศ. 1996 ลินุกซ์ เวอร์ชัน 2.0 เพิ่มเติมความสามารถในการสนับสนุนสถาปัตยกรรมหลายรูปแบบ (รวม 64 บิตของ Alpha) และสนับสนุนสถาปัตยกรรมมัลติโปรเซสเซอร์ ทำงานได้บน Mach Microkernel, พีซี และPowerMac นอกจากนี้ยังปรับปรุงคุณสมบัติในการจัดการหน่วยความจำ เพิ่มประสิทธิภาพของระบบไฟล์, หน่วยความจำเสมือน เพิ่มเติมเน็ตเวิร์คโพรโทคอล เช่น Apple Talk, AX.25 และ ISDN เป็นต้น

ปี ค.ศ. 1999 ลินุกซ์ เวอร์ชัน 2.2 เพิ่มเติมพอร์ตสำหรับ UltraSparc ทางด้านเน็ตเวิร์คมีการขยายขีดความสามารถทำให้ไฟล์วอลล์มีความยืดหยุ่นมากขึ้น มีการจัดเส้นทางและการจราจรของเส้นทางได้ดีขึ้น kernel ได้ถูกพัฒนามาอย่างต่อเนื่องอย่างไม่หยุดยั้ง ปัจจุบัน ลินุกซ์ kernel อยู่ที่เวอร์ชันที่ 2.6

เนื่องจากผู้ใช้สามารถพัฒนา ลินุกซ์ ได้ ทำให้มีการพัฒนาออกไปหลายกลุ่ม ลินุกซ์ จึงมีหลายตระกูล(Distribution หรือ Distros) ในปัจจุบัน เช่น Debian, OpenSUSE, Slackware, RedHat, CentOS, Ubuntu, Mint เป็นต้น

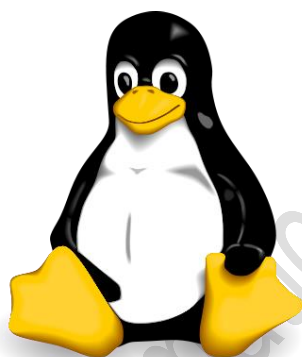
คุณสมบัติเด่นของยูนิกซ์และลินุกซ์

- เป็นระบบปฏิบัติการที่สามารถใช้งานได้ฟรี เนื่องจาก ลินุกซ์ เป็น free software
- เป็นซอฟต์แวร์โอเพนซอร์ส(ระบบปฏิบัติการแบบเปิด) เคอร์เนล (Kernel) หรือแก่นซึ่งเป็นส่วนที่สำคัญของระบบปฏิบัติการของและซอฟต์แวร์ส่วนใหญ่ของ ลินุกซ์ ถูกแจกจ่ายออกไปอย่างแพร่หลายทั้งตัวโปรแกรมและซอร์สโค้ด (ซึ่งส่วนมากพัฒนาด้วยภาษา C)

ดังนั้นทำให้นักคอมพิวเตอร์ทั่วโลกสามารถนำซอร์สโค้ดต้นแบบที่ถูกแจกจ่ายมาให้พัฒนาซอฟต์แวร์ได้อย่างอิสระ ในสถานศึกษาต่างๆ ทั่วโลกนิยมใช้เป็นตันต้นแบบในการเรียนการสอนวิชาระบบปฏิบัติการ เพื่อให้นักศึกษาเข้าใจได้ง่าย นำไปสู่การพัฒนา kernel , ดีไวซ์ ไดร์เวอร์ ตลอดจนแอพพลิเคชันต่างๆ ได้ง่าย

- มีความสามารถในการทำงานร่วมกับระบบปฏิบัติการยูนิกซ์ได้เป็นอย่างดี (UNIX Compatible) เนื่องจากลินุกซ์ถูกพัฒนามาจากยูนิกซ์นั่นเอง จึงมีคุณสมบัติที่มีความน่าเชื่อถือและความเสถียรของระบบสูง เป็นระบบปฏิบัติการแบบมัลติยูเซอร์ และมัลติแทสกิ้ง และสามารถใช้งานในรูปแบบกราฟิกโดยใช้ระบบ X Window ที่สนับสนุนโปรแกรม Window Manager หลายตัว
- สามารถทำงานได้กับสถาปัตยกรรมของเครื่องคอมพิวเตอร์ได้หลายตระกูล เช่น ซีพียู 80386 ของอินเทล, Motorola 680x0, Alpha, PowerPC และ SPARC เป็นต้น ลินุกซ์ยังมีความสามารถสนับสนุนอุปกรณ์ต่อพ่วงมากมาย เช่น การ์ดแสดงผล ซีดีรอม เครื่องพิมพ์ ฮาร์ดดิสก์ และ เน็ตเวิร์คการ์ด ลินุกซ์สนับสนุนระบบบัสได้หลายแบบ เช่น EISA, ISA, VESA Localbus และPCI เป็นต้น
- มีความสามารถในการทำงานร่วมกับระบบปฏิบัติการดอส(Dos) และวินโดวส์(Windows) ลินุกซ์สามารถติดตั้งบนฮาร์ดดิสก์ตัวเดียวกันกับดอสและวินโดวส์ได้ โดยการแบ่งพาร์ติชันเพิ่มเติม ลินุกซ์สามารถอ่านและเขียนฮาร์ดดิสก์ที่เป็นโครงสร้างไฟล์แบบดอสหรือวินโดวส์ได้ และยังสนับสนุนโครงสร้างไฟล์ของระบบปฏิบัติการอื่นๆ ได้ด้วย เช่น Windows 98/ME (FAT 32), Windows NT/2000/2003/2008(NTFS), Netware, OS/2 (HPFS), MINIX, NFS เป็นต้น ในปัจจุบันสามารถนำโปรแกรมที่ทำงานบนดอสหรือวินโดวส์มาทำงานบนลินุกซ์ได้เกือบทุกโปรแกรม(ทำงานผ่านโปรแกรม wine โดยการจำลองโปรแกรมบนวินโดวส์มาทำงานบนลินุกซ์)
- สนับสนุนการทำงานด้านเครือข่ายแบบเต็มรูปแบบ เช่นสามารถรองรับการเชื่อมต่อระบบเครือข่ายแบบ Ethernet, Token Ring, SLIP, PPP, ISDN, Frame Relay, ATM มีความสามารถทำงานด้านโพรโทคอลเราติ้งด้วย เช่น RIP, OSPF, BGP, Multicast, IPV6 เป็นต้น รวมทั้งยังสามารถใช้ลินุกซ์เป็นอินเตอร์เน็ตเซร์ฟเวอร์ด้วย เพื่อให้บริการในอินเตอร์เน็ต เช่น Firewall, DNS, DHCP, FTP, Telnet, NNTP, SMTP, Gopher และ WWW เป็นต้น
- เคอร์เนลของระบบปฏิบัติการยูนิกซ์/ลินุกซ์ มีความยืดหยุ่นในการใช้งานสูง สามารถใช้กับงานเล็กๆ ไปจนถึงงานที่มีขนาดใหญ่ได้ และมีสามารถทำงานที่ซับซ้อนได้ดีกว่าระบบปฏิบัติการอื่นๆ

- มีความสามารถในการมีการใช้ไลบรารีไฟล์ร่วมกัน (Dynamically Linked Shared Libraries) ซึ่คุณสมบัติดังกล่าวนี้ทำให้โปรแกรมที่นำมาทำงานบนลินุกซ์มีขนาดที่เล็กลง สามารถทำงานได้เร็วขึ้น ส่งผลให้ระบบปฏิบัติการมีขนาดที่เล็กลงมาก (มีรูปแบบคล้ายการทำงานของ dll ไฟล์ของระบบปฏิบัติการวินโดวส์
- การช่วยเหลือเมื่อเกิดปัญหาจากการใช้งาน เนื่องจากเป็นซอฟต์แวร์แบบเปิด(Opensource) จึงมีผู้พัฒนาเป็นจำนวนมาก เมื่อเกิดปัญหาขึ้นจากการใช้งาน สามารถขอความช่วยเหลือได้ตลอดเวลาผ่านหลายทาง เช่น New Group, Mailling List หรือเว็บไซต์มากมายทั่วโลกที่มีเว็บบอร์ดต่างๆ ที่ให้คำแนะนำการใช้งาน



Linux Logo

สายพันธุ์ลินุกซ์ (Linux Distributions) [5-6]

เนื่องจากลินุกซ์มีพื้นฐานมาจาก Open source ทำให้มีผู้สร้างลินุกซ์สายพันธุ์ใหม่ๆ ขึ้นมาเป็นจำนวนมาก ซึ่งนิยมเรียกว่า Distributions (distro) ปัจจุบันมีลินุกซ์เกิดขึ้นมาใหม่มากกว่า 180 สายพันธุ์(ค.ศ. 2010) และมีการจัดอันดับความนิยมของลินุกซ์ [CH1-07] ในที่นี้จะนำเสนอ 10 distro ยอดนิยมคือ

ลำดับที่	Distributions	Website	Logo
1	Ubuntu	http://distrowatch.com/ubuntu	ubuntu
2	Fedora	http://distrowatch.com/fedora	fedora

3	Mint	http://distrowatch.com/mint	
4	openSUSE	http://distrowatch.com/suse	
5	Debian	http://distrowatch.com/debian	
6	Mandriva	http://distrowatch.com/mandriva	
7	Puppy	http://distrowatch.com/puppy	
8	PCLinuxOS	http://distrowatch.com/pclinuxos	
9	Sabayon	http://distrowatch.com/sabayon	
10	Arch	http://distrowatch.com/arch	

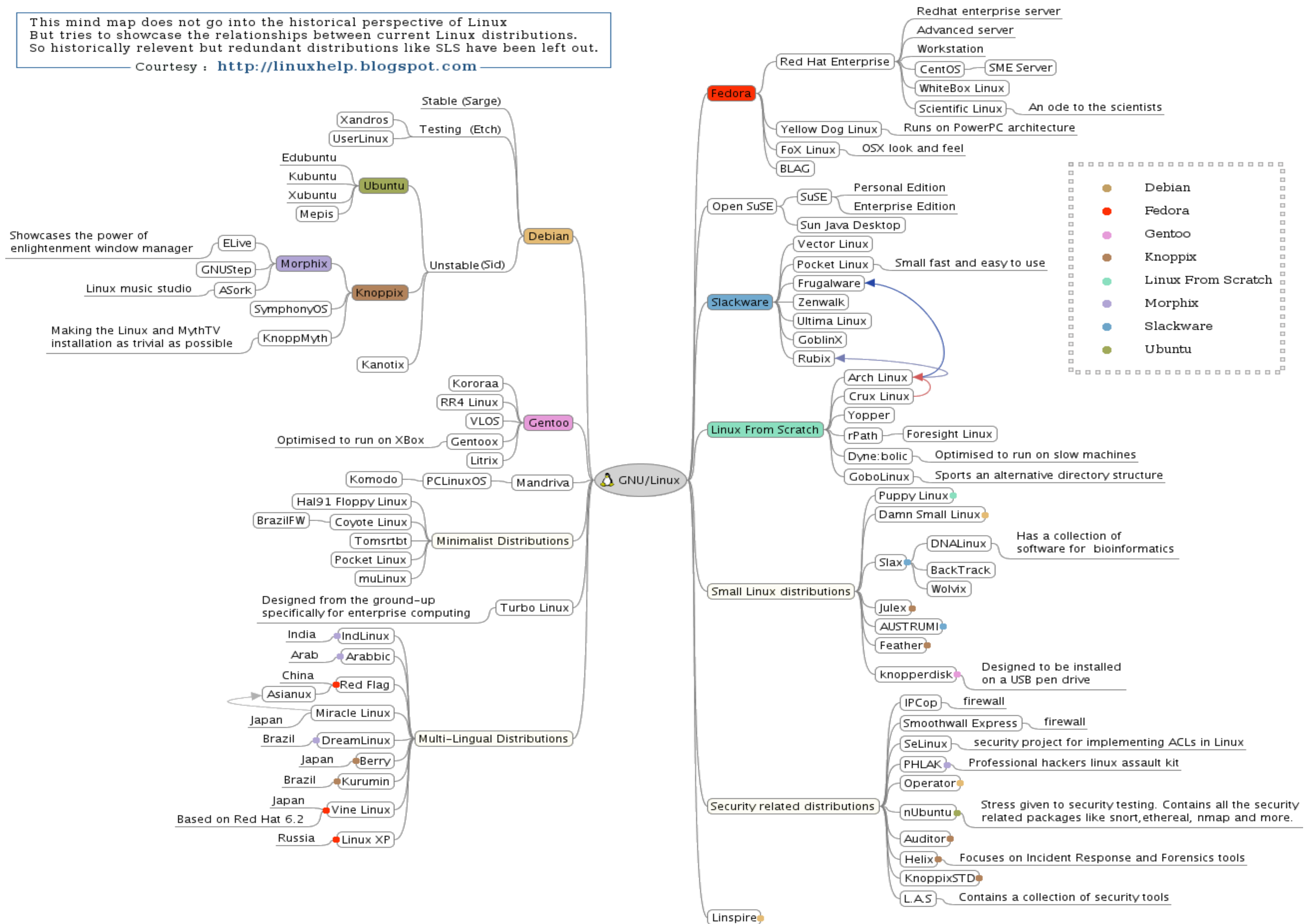
Logo distributions อื่นๆ และยูนิกซ์ค่ายต่างๆ


ubuntu
SUSE
Mandriva
fedora
KNOPPIX
MEPIS
redhat
solaris
gentoo linux
PCLinuxOS
Novell
FreeBSD
xandros
LUNAR
aurox
slackware
kubuntu
Vector
KANOTIX
CentOS
debian
slax
aLinux
xubuntu


redhat
debian
NetBSD
solaris
ubuntu

สรุปท้ายบท

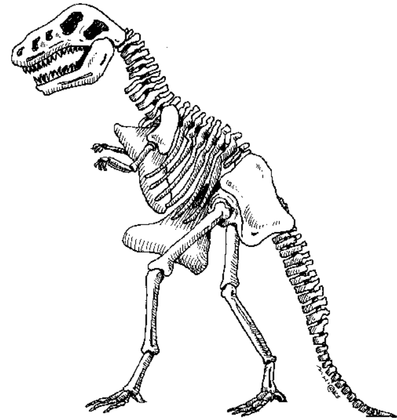
ยูนิกส์เป็นระบบปฏิบัติการแรก ๆ ที่เกิดขึ้นในโลก เพื่อใช้ควบคุมเครื่องคอมพิวเตอร์ให้สามารถทำงานได้เกิดประสิทธิภาพสูงสุด และรวดเร็วขึ้นสามารถทำงานได้หลายๆ งานพร้อมๆ กัน ผู้ใช้สามารถเข้าใช้งานระบบปฏิบัติการดังกล่าวได้พร้อมกันหลายคน โดยเริ่มพัฒนาในปี 1957 ในห้องปฏิบัติการ AT&T Bell Labs โดย Ken Thompson, Dennis Ritchie และ Douglas McIlroy ต่อจากนั้นก็ได้ร่วมมือกับ สถาบันเทคโนโลยีแมสซาชูเซตส์ (MIT) พัฒนาความสามารถให้เพิ่มขึ้น จากนั้นได้มีการพัฒาต่อยอดมาเรื่อยๆ อย่างไม่หยุดยั้งเกิดเป็นยูนิกส์สายพันธ์ต่างๆ ขึ้นมาอย่างมากมาย เช่น BSD, Solaris, DEC เป็นต้น ในปี 1987 Andrew S.Tanenbaum ได้ออกแบบสร้าง ยูนิกซ์สำหรับเครื่องไมโครคอมพิวเตอร์ ซึ่งสามารถทำงานได้ทั้งบนเครื่อง PC, Mac, Amiga โดยให้ชื่อว่า Minix และยังแจกซอร์สโค้ดฟรีให้แก่นักวิจัยเพื่อนำไปพัฒนาต่อ จึงเป็นแรงบันดาลใจให้ ลินุส โตร์วัลดส์ (Linus Torvalds) เอาซอร์สโค้ดดังกล่าวมาเป็นต้นแบบในการพัฒนาต่อ ในเวลาต่อมาจึงเรียกชื่อใหม่ว่า ลินุกซ์ ซึ่งภายหลังได้รับความนิยมสูงมากเนื่องจากระบบปฏิบัติการดังกล่าวอยู่ภายใต้ลิขสิทธิ์แบบ GNU ทำให้เกิดสายพันธ์ต่างๆ มากมาย



This mind map does not go into the historical perspective of Linux
But tries to showcase the relationships between current Linux distributions.
So historically relevent but redundant distributions like SLS have been left out.
Courtesy : http://linuxhelp.blogspot.com
GNU/Linux
Debian
Stable (Sarge)
Testing (Etch)
Xandros
UserLinux
Unstable (Sid)
Ubuntu
Edubuntu
Kubuntu
Xubuntu
Mepis
Knoppix
Morphix
ELive
Showcases the power of enlightenment window manager
GNUStep
ASork
Linux music studio
SymphonyOS
KnoppMyth
Making the Linux and MythTV installation as trivial as possible
Kanotix
Gentoo
Kororaa
RR4 Linux
VLOS
Gentoox
Optimised to run on XBox
Litrix
Mandriva
PCLinuxOS
Komodo
Minimalist Distributions
Hal91 Floppy Linux
Coyote Linux
BrazilFW
Tomsrtbt
Pocket Linux
muLinux
Turbo Linux
Designed from the ground-up specifically for enterprise computing
Multi-Lingual Distributions
IndLinux
India
Arabbic
Arab
Red Flag
China
Asianux
Miracle Linux
Japan
DreamLinux
Brazil
Berry
Japan
Kurumin
Brazil
Vine Linux
Japan
Based on Red Hat 6.2
Linux XP
Russia
Fedora
Red Hat Enterprise
Redhat enterprise server
Advanced server
Workstation
CentOS
SME Server
WhiteBox Linux
Scientific Linux
An ode to the scientists
Yellow Dog Linux
Runs on PowerPC architecture
FoX Linux
OSX look and feel
BLAG
Open SuSE
SuSE
Personal Edition
Enterprise Edition
Sun Java Desktop
Slackware
Vector Linux
Pocket Linux
Small fast and easy to use
Frugalware
Zenwalk
Ultima Linux
GoblinX
Rubix
Linux From Scratch
Arch Linux
Crux Linux
Yopper
rPath
Foresight Linux
Dyne:bolic
Optimised to run on slow machines
GoboLinux
Sports an alternative directory structure
Small Linux distributions
Puppy Linux
Damn Small Linux
Slax
DNALinux
Has a collection of software for bioinformatics
BackTrack
Wolvix
Julex
AUSTRUMI
Feather
knopperdisk
Designed to be installed on a USB pen drive
Security related distributions
IPCop
firewall
Smoothwall Express
firewall
SeLinux
security project for implementing ACLs in Linux
PHLAK
Professional hackers linux assault kit
Operator
nUbuntu
Stress given to security testing. Contains all the security related packages like snort, ethereal, nmap and more.
Auditor
Helix
Focuses on Incident Response and Forensics tools
KnoppixSTD
L.A.S
Contains a collection of security tools
Linspire
Debian
Fedora
Gentoo
Knoppix
Linux From Scratch
Morphix
Slackware
Ubuntu

คำถามท้ายบท

1. ระบบปฏิบัติการคืออะไร?
2. ระบบปฏิบัติการมีหน้าที่อะไรบ้าง
3. ระบบปฏิบัติการยูนิกซ์และลินุกซ์แตกต่างกันอย่างไร
4. ยูนิกซ์เกิดขึ้นครั้งแรก เนื่องด้วยเหตุผลอะไร
5. ผู้ที่คิดค้นและสร้างระบบปฏิบัติการยูนิกซ์คือใคร
6. กลุ่มคนที่ร่วมกันพัฒนายูนิกซ์กลุ่มแรกคือใครบ้าง
7. ยูนิกซ์ในยุคแรกพัฒนาด้วยภาษาอะไร
8. ภาษาที่เป็นที่นิยมในการพัฒนายูนิกซ์คืออะไร เนื่องมาจากสาเหตุอะไร
9. ข้อดีและข้อเสียของระบบปฏิบัติการยูนิกซ์มีอะไรบ้าง
10. ใครคือผู้ที่เริ่มต้นสร้างระบบปฏิบัติการลินุกซ์เป็นคนแรก
11. ลินุกซ์ได้รับการพัฒนามาจากระบบปฏิบัติการชื่อว่าอะไร และใครเป็นผู้ที่พัฒนาระบบปฏิบัติการต้นแบบนั้น
12. จุดเด่นของระบบปฏิบัติการลินุกซ์ที่มีความแตกต่างจากยูนิกซ์คืออะไร
13. ลินุกซ์สามารถทำงานได้บนสถาปัตยกรรมอะไรบ้าง
14. ใครเป็นบุคคลที่ก่อตั้งโครงการกนูและก่อตั้งขึ้นเพื่อวัตถุประสงค์อะไร
15. จงบอกสาเหตุที่ทำให้ระบบปฏิบัติการลินุกซ์ได้รับความนิยมอย่างรวดเร็ว



บทที่

2

โครงสร้างของยูนิกซ์และลินุกซ์

วัตถุประสงค์

1. สามารถอธิบายองค์ประกอบของระบบปฏิบัติการว่ามีกี่ส่วนอะไรบ้าง และแต่ละส่วนทำงานอย่างไร
2. สามารถอธิบายความหมายของเคอร์เนล(kernel) คืออะไร และทำหน้าที่อะไรในระบบปฏิบัติการ
3. สามารถอธิบายของเชลล์ (shell) และหน้าที่ของเชลล์
4. สามารถอธิบายถึง X-Window, Window Manager, X Desktop
5. สามารถอธิบายระบบไฟล์ระบบปฏิบัติการบนลินุกซ์
6. สามารถอธิบายโครงสร้างไดเรคทอรีและไฟล์บนลินุกซ์ประกอบไปด้วยอะไรบ้าง และแต่ละส่วนทำหน้าที่อย่างไร

บทที่ 2
โครงสร้างของยูนิกซ์และลินุกซ์

บทนำ

ในบทนี้จำกล่าวถึงโครงสร้างของระบบปฏิบัติการยูนิกซ์และลินุกซ์ ซึ่งประกอบไปด้วย 4 ส่วนคือ ฮาร์ดแวร์ เคอร์เนล เชลล์ และยูทิลิตี้กับซอฟต์แวร์ประยุกต์ ต่อจากนั้นจะกล่าวถึงโครงสร้างของไฟล์และไดเรคทรอรีบนยูนิกซ์และลินุกซ์

โครงสร้างของยูนิกซ์และลินุกซ์แบ่งออกได้เป็น 4 ส่วนคือ [8]

- **ฮาร์ดแวร์ (Hardware)** หมายถึง อุปกรณ์ต่างๆ ที่ประกอบขึ้นเป็นเครื่องคอมพิวเตอร์ มีลักษณะเป็นโครงร่างสามารถมองเห็นด้วยตาและสัมผัสได้ (รูปธรรม) เช่น หน่วยประมวลผลกลาง หน่วยความจำ เมนบอร์ด จอภาพ คีย์บอร์ด เครื่องพิมพ์ เมาส์ เป็นต้น
- **เคอร์เนล (kernel)** เคอร์เนลเป็นส่วนหนึ่งของซอฟต์แวร์ในระบบปฏิบัติการที่สำคัญ เรียกได้ว่าเป็นแกนหรือหัวใจของระบบก็ว่าได้ เคอร์เนลจะมีหน้าที่ควบคุมการทำงานทั้งหมดของระบบ ตั้งแต่การจัดสรรทรัพยากรของระบบบริการ โพรเซสงาน (Process) การจัดการไฟล์และอุปกรณ์อินพุต , เอาต์พุต บริหารหน่วยความจำ โดยเคอร์เนลจะควบคุมอุปกรณ์ฮาร์ดแวร์ของเครื่องทั้งหมด ดังนั้นเคอร์เนลจึงขึ้นอยู่กับฮาร์ดแวร์ ถ้าฮาร์ดแวร์เปลี่ยนรุ่นใหม่ เคอร์เนลก็จะต้องเปลี่ยนไปด้วยเคอร์เนล เคอร์เนลนิยมเขียนขึ้นด้วยภาษาแอสเซมบลี้หรือภาษาซี และเป็นส่วนที่ขึ้นอยู่กับฮาร์ดแวร์ของเครื่อง (hardware dependent) ด้วย นั่นคือ ถ้าโครงสร้างทางฮาร์ดแวร์ของเครื่องมีการเปลี่ยนแปลง ส่วนของเคอร์แนลต้องถูกนำมาแก้ไขใหม่ด้วยเพื่อให้สามารถทำงานกับฮาร์ดแวร์ชิ้นใหม่ได้
- **เชลล์ (shell)** เป็นซอฟต์แวร์ตัวกลางที่ทำหน้าที่ติดต่อระหว่างผู้ใช้กับเคอร์เนล แบ่งออกเป็น 2 กลุ่มคือ กลุ่มเชลล์ที่มีการติดต่อกับผู้ใช้ด้วยคำสั่ง (Command Line Interface - CLI) และเชลล์ที่มีการติดต่อกับผู้ใช้โดยใช้ภาพสัญลักษณ์ (Graphic User Interface - GUI)
    - CLI ผู้ใช้จะต้องป้อนคำสั่งผ่านทางคีย์บอร์ดเป็นลักษณะข้อความ(หรือเรียกว่า Text Shell ก็ได้) ที่เหมาะสมและถูกต้อง ดังนั้นผู้ใช้จะต้องจดจำรายละเอียดต่างๆ ของคำสั่ง ยิ่งจดจำได้มากก็จะสามารถใช้งานในแบบ CLI ได้สะดวกมากขึ้น CLI นิยมใช้ในระบบปฏิบัติการยูนิกซ์และลินุกซ์ เนื่องจาก

เชลล์ดังกล่าวใช้ทรัพยากรของระบบน้อย และเชลล์ก็มีให้เลือกใช้งานได้หลายตัว เช่น Bourne shell, C-shell, และ Korn shell เป็นต้น แต่ละเชลล์แบบมีความสามารถพื้นฐานคล้ายกัน แต่มีขีดความสามารถในการเขียนเชลล์สคริปต่างกัน

- กราฟฟิกเชลล์(Graphic shell) การเชื่อมต่อกับผู้ใช้ด้วยภาพหรือสัญลักษณ์ ได้รับความนิยมแพร่หลายมากในปัจจุบัน เช่น วินโดวส์รุ่นต่างๆ สำหรับระบบปฏิบัติการยูนิกซ์และลินุกซ์มีการพัฒนาการเชื่อมต่อกับผู้ใช้ด้วยลักษณะของกราฟฟิกเช่นเดียวกัน เรียกว่าระบบ X-Window [9] ซึ่งเป็นระบบที่มีการพัฒนามาเป็นระยะเวลานาน และมีโปรแกรมจัดการ Window ให้เลือกใช้หลายตัวเช่น OSF/Motif, FVM, KDE เป็นต้น และมีโปรแกรมประยุกต์จำนวนหนึ่งสำหรับใช้งานในระบบเหล่านี้ คือ

X-Windows

โดยปกติการใช้งานของยูนิกซ์ จะต้องพิมพ์คำสั่ง (CLI) เพื่อสั่งงาน ถ้าเปรียบเทียบการใช้งานกับวินโดวส์แล้ว ยูนิกซ์มีการใช้งานที่ค่อนข้างลำบากกว่า เนื่องจากต้องจำคำสั่งต่าง ๆ มากมาย ดังนั้นจึงมีผู้ที่พยายามทำให้ยูนิกซ์มีการใช้งานที่ง่ายขึ้น เช่นเดียวกับวินโดวส์

จากการสนับสนุนของ DEC ได้มีการเริ่มพัฒนาต่อจาก W-Window ซึ่งคิดค้นโดย Robert Scheifler และพัฒนาต่อมาถึง X-Window ส่วนสำคัญที่ทำให้ X-Window เป็นที่แพร่หลายมาก นอกเหนือจากความสะดวกในการใช้งานแล้ว นั่นคือ เป็นโปรแกรมที่แจกฟรี และมีโปรแกรมอื่น ๆ ที่สนับสนุนการใช้งานอื่น ๆ มากมาย สำหรับหน้าตาของ X-Window นี้จะมีลักษณะและการใช้งานเช่นเดียวกันกับ Microsoft Windows

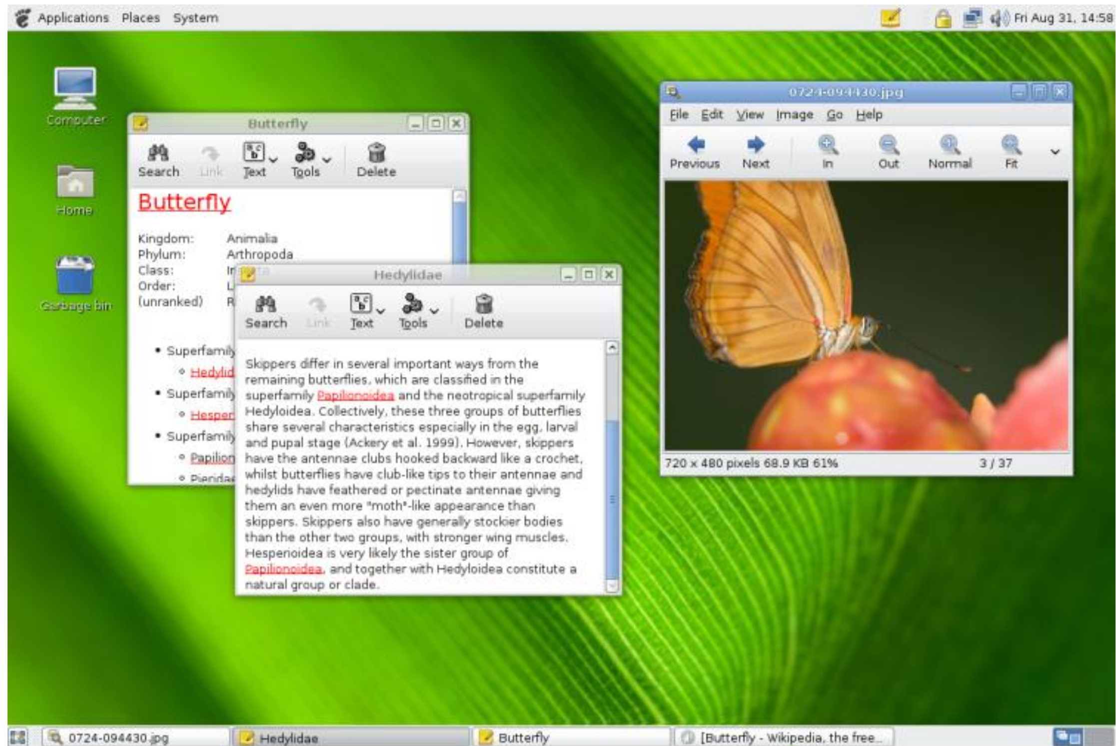
Window Manager

นำเอารูปภาพหรือไอคอนต่าง ๆ ที่ช่วยในการติดต่อสื่อสารระหว่างผู้ใช้กับระบบ X-Window ซึ่งเป็นตัวช่วยในการทำงานให้เกิดความสะดวกมากยิ่งขึ้น การใช้งานจะอยู่ในลักษณะที่เราเรียกว่า กราฟฟิกโหมด แทนการพิมพ์คำสั่งหรือ command line ในระบบ เท็กโหมด (โหมดตัวอักษร) สำหรับ Window Manager ที่เป็นที่รู้จักกันดีคือ FVWM และ FVWM2

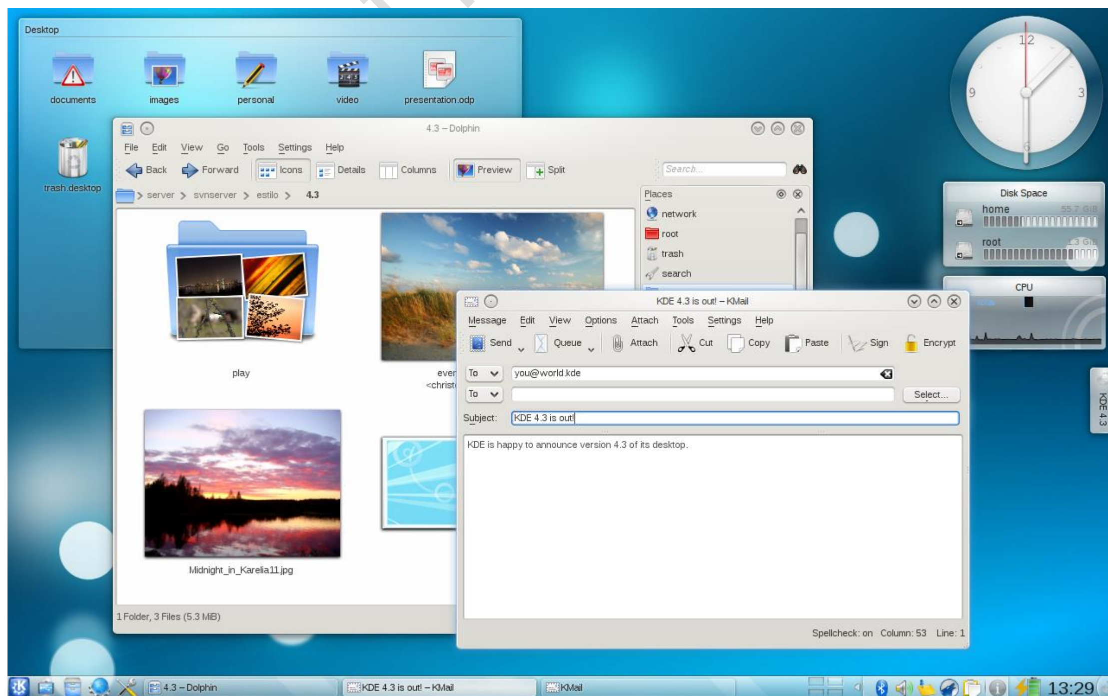
X Desktop Environment

พัฒนาการต่อจาก Window Manager ซึ่งตัวโปรแกรมเองจะทำงานครอบ Window Manager อีกทีหนึ่ง ซึ่งทำหน้าที่จัดการเกี่ยวกับ ไฟล์ รวมทั้งตัว

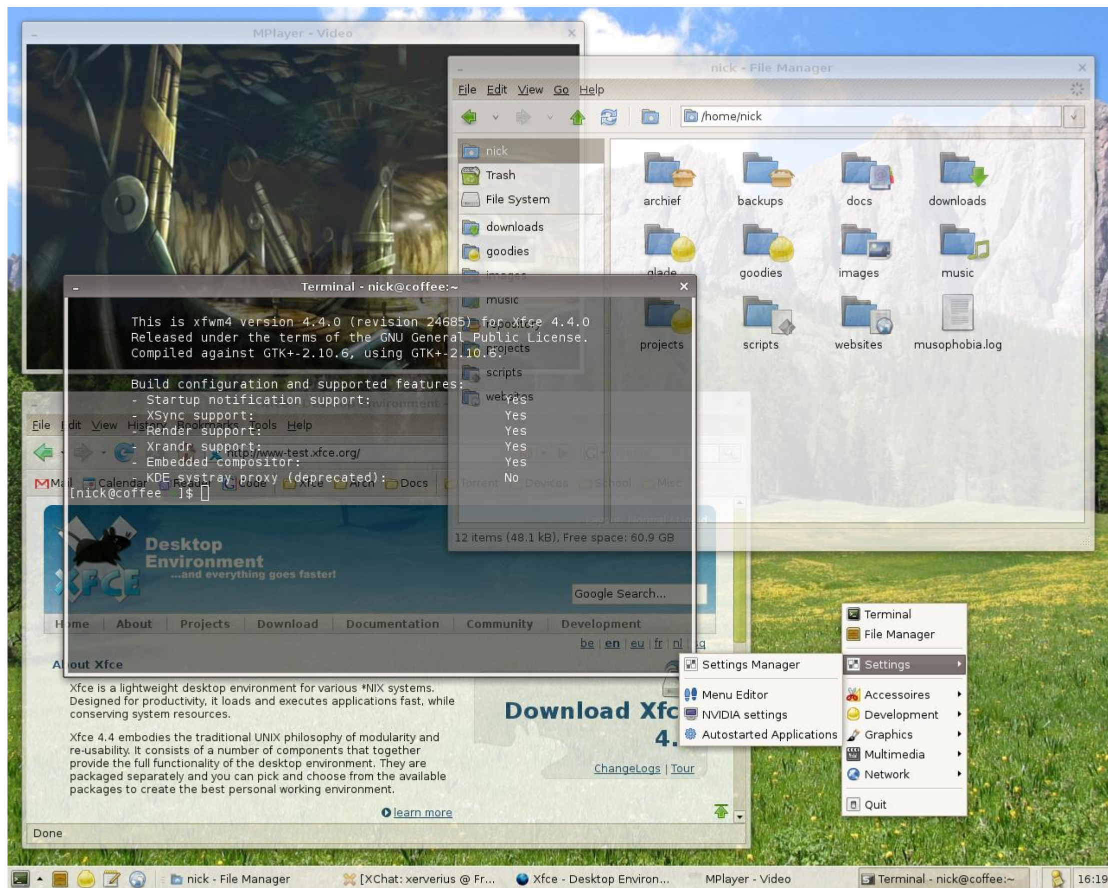
Window Manager ด้วย สำหรับ X Desktop Environment ที่นิยมใช้กันคือ KDE (หน้าตาและรูปร่างคล้ายกับ Microsoft Windows)) และ GNOME KDE (หน้าตาและรูปร่างคล้ายกับ KDE แต่ใช้งานได้ง่ายกว่า)



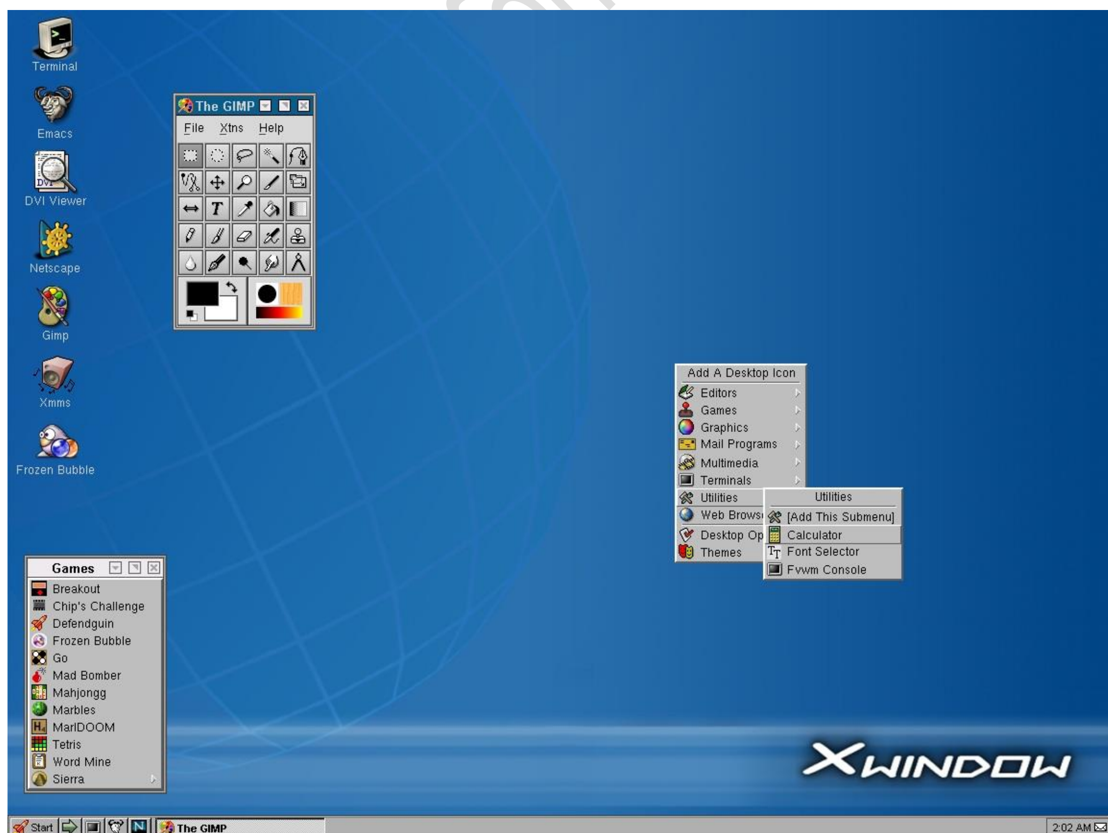
GNOME X Desktop Environment



KDE X Desktop Environment



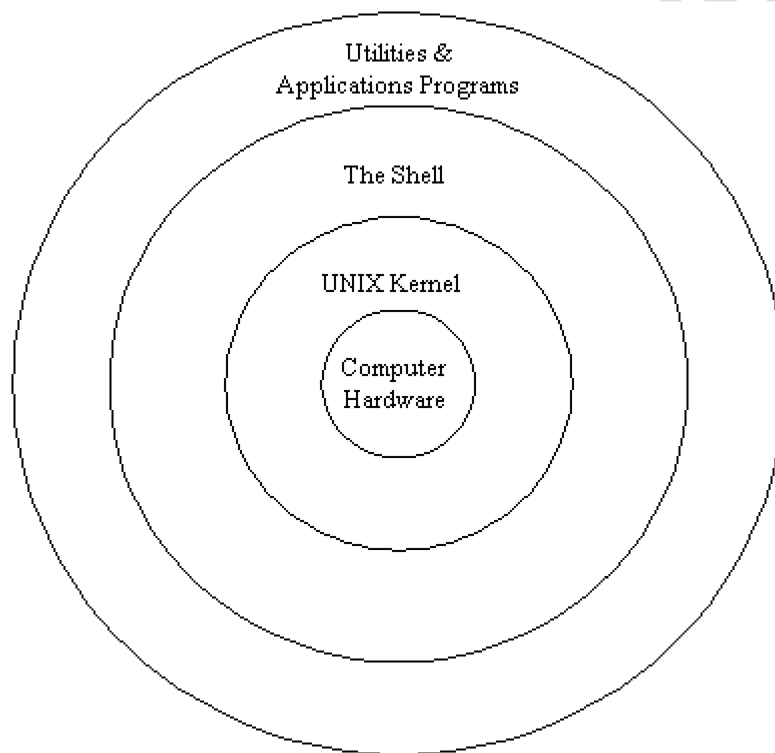
Xfce X Desktop Environment

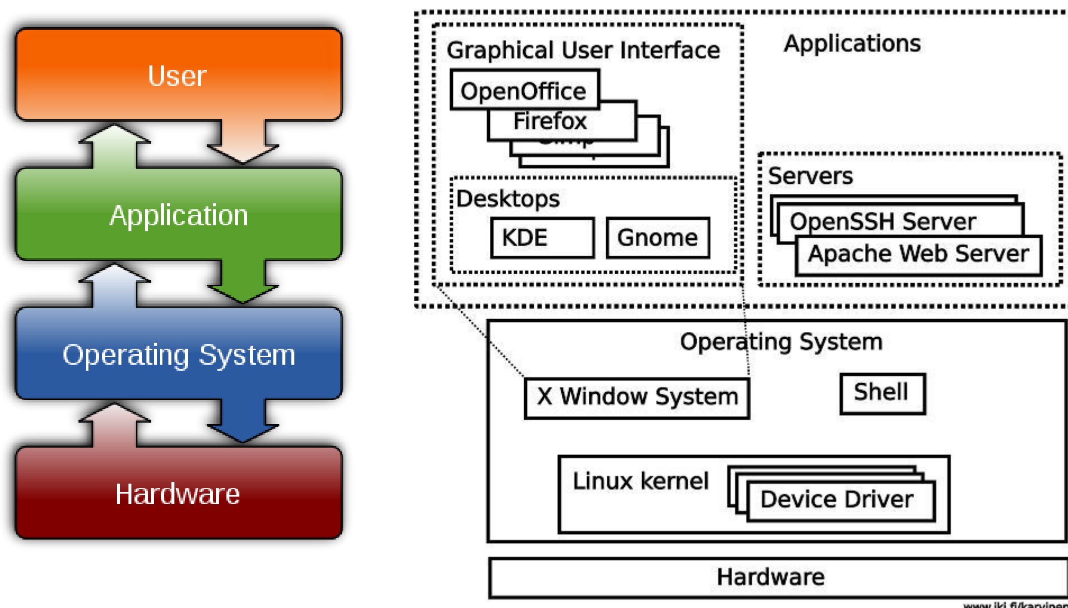


FVWM Window Manager

- **ยูทิลลิตี้และซอฟต์แวร์ประยุกต์ (Utilities and Application Programs)** เป็นส่วนที่ทำหน้าที่จัดการระบบและอำนวยความสะดวกให้กับผู้ใช้งาน ประกอบไปด้วยโปรแกรมที่ทำหน้าที่จัดการระบบในส่วนต่าง ๆ เช่น จัดการระบบไฟล์ ส่วนผู้ใช้งาน ระบบรักษาความปลอดภัย และระบบเครือข่าย เป็นต้น

ระบบปฏิบัติการลินุกซ์จะใช้ทรัพยากรน้อยกว่ายูนิกซ์ เนื่องจากลินุกซ์เป็นระบบปฏิบัติการที่ออกแบบสำหรับให้ทำงานบนเครื่องพีซีที่ที่ใช้สถาปัตยกรรมแบบ x86 จึงสามารถนำฮาร์ดแวร์หรือเครื่องไมโครคอมพิวเตอร์ที่มีความสามารถต่ำมาทำการติดตั้งได้ และสามารถแสดงผลแบบกราฟฟิกได้





โครงสร้างของระบบปฏิบัติการยูนิกซ์และลินุกซ์

ประเภทของไฟล์บนระบบปฏิบัติการยูนิกซ์และลินุกซ์ [10-11]

โครงสร้างของไฟล์บนระบบปฏิบัติการยูนิกซ์คล้ายกับลินุกซ์ ดังนั้นในส่วนนี้จะกล่าวถึงโครงสร้างของไฟล์บนลินุกซ์เป็นหลัก ในลินุกซ์ ทำการเก็บข้อมูลโดยใช้ ไฟล์ และไดเรกทอรี โดยจะมีลักษณะเป็นรูปแบบลำดับชั้น (hierarchy) หรือโครงสร้างแบบต้นไม้ ไดเรกทอรีจะเปรียบเสมือนแฟ้ม ที่สามารถเก็บไฟล์ต่างๆ ไดเรกทอรีลำดับบนสุดจะถูกเรียกว่า ไดเรกทอรีราก (root directory) ระบบไฟล์ที่ใช้บนลินุกซ์ถูกพัฒนาขึ้นครั้งแรกคือระบบไฟล์แบบมินิกซ์ (Minix FS) ซึ่งระบบไฟล์แบบมินิกซ์มีข้อจำกัดคือ บล็อกแอดเดรส (Block Address) มีขนาดใหญ่สุดเพียง 16 บิต ด้วยเหตุนี้จึงทำให้ขนาดสูงสุดของไฟล์จำกัดอยู่ที่ 64 (2^{16} = 65,535) เมกกะไบต์และตั้งชื่อไฟล์ได้ยาวสูงสุดได้เพียง 14 ตัวอักษร จึงได้มีการออกแบบและพัฒนาระบบไฟล์ขึ้นมาใหม่คือระบบไฟล์แบบ Ext FS (Extended File System) ที่ช่วยลดข้อจำกัดเหล่านี้ โดยบล็อกแอดเดรสมีขนาดใหญ่เพิ่มขึ้นเป็น 2 กิกะไบต์ และสามารถตั้งชื่อไฟล์ได้สูงสุดเป็น 255 ตัวอักษร แต่ระบบไฟล์แบบเอ็กซ์เทนยังไม่สนับสนุนการแก้ไข ไอโหนด (Inode) การแก้ไข Timestamps และการดำเนินการของระบบไฟล์ใช้ลิสต์แบบไม่เรียงลำดับ จึงได้มีการออกแบบและพัฒนาระบบไฟล์ขึ้นมาใหม่ 2 ระบบไฟล์เพื่อแก้ไขปัญหาคือ ระบบไฟล์แบบ Xia (Xia File System) และระบบไฟล์แบบ Ext2 FS (Second Extended File System) ในปัจจุบันระบบปฏิบัติการลีนุกซ์ได้มีการพัฒนาระบบไฟล์ขึ้นอีก 2 แบบคือ Ext3 และ Ext4 ซึ่ง Ext3 [12] มีคุณสมบัติที่เพิ่มขึ้นหลายประการจาก Ext 2 แต่ที่หลักๆ คือ เป็นระบบไฟล์แบบ journaling เมื่อข้อมูลในระบบเสียหาย ไม่จำเป็นต้องมีการใช้คำสั่ง fsck เพื่อตรวจสอบและซ่อมแซมแฟ้มข้อมูลนั้น เพราะว่าการทำงานของ journaling นั้นจะมีการกู้แฟ้มข้อมูลที่เสียหายขึ้นมาให้โดยอัตโนมัติ รองรับพื้นที่ได้มากสุด 16 เทอราไบต์

(TB) และขนาดของไฟล์ใหญ่ที่สุด 2 TB มีไดเรกทอรีย่อยๆ ได้มากถึง 32,000 ไดเรกทอรีย่อย ขณะเขียนไฟล์ก็จะมีการจัดสรรพื้นที่ทีละบล็อค (only allocates one block) ในแต่ละครั้งที่เขียนลงไฟล์ ก็คือว่าถ้ามีการเขียนไฟล์ขนาด 1 MB ก็จะมีการจัดสรรบล็อคในการเขียนไฟล์ 256 ครั้ง (block size = 4KB) สำหรับ Ext4 ได้ปรับปรุงแก้ไขส่วนสำคัญในเรื่องของโครงสร้างของระบบไฟล์ใน Ext3 เพิ่มขึ้น ตัวอย่างเช่น ได้มีการกำหนดพื้นที่ที่เก็บข้อมูลไฟล์ไว้ล่วงหน้า ทำให้ระบบไฟล์มีการจัดสรรพื้นที่เก็บข้อมูลที่ดีขึ้น เพิ่มความน่าเชื่อถือมากขึ้น สรุปได้ดังนี้ Ext4 [13] ได้มีการเพิ่ม block address เป็นขนาด 48-bit ทำให้สามารถรองรับพื้นที่ได้ถึง 1 EB (1 EB = 1,048,576 TB) และขนาดของไฟล์ใหญ่ที่สุด 16 TB (1 EB = 1024 PB, 1 PB = 1024 TB, 1 TB = 1024 GB เรียกว่า Large file system/file sizes) มีไดเรกทอรีย่อยได้ถึง 64,000 ไดเรกทอรีย่อย(Sub Directory) มีการจองพื้นที่ที่อยู่ติดกัน ก่อนที่จะเขียนไฟล์ ทำให้ไฟล์ที่มีขนาดใหญ่มีประสิทธิภาพมากขึ้นและช่วยลดการกระจายของข้อมูล(Extents) ใช้ "multiblock allocator" (mballoc) ซึ่งจะมีการจัดสรรบล็อกได้ทีละหลายๆบล็อก ในการเรียกเขียนไฟล์ใน 1 ครั้ง(Multiblock allocation) ใช้เทคนิคที่เรียกว่า allocate-on-flush หรือที่รู้จักกันในชื่อ delayed allocation ซึ่งวิธีของ delayed allocation นี้ ถ้าเกิดมีการเขียนไฟล์ขึ้นมามันจะยังไม่จัดสรรพื้นที่ของบล็อกที่จะเขียนโดยทันที จนกว่าข้อมูลนั้นจะถูกเขียนลงดิสก์จริงๆ จึงจะมีการจัดสรรพื้นที่ของบล็อก(delayed allocation) เพิ่มความเร็วในการตรวจสอบและซ่อมแซมแฟ้มข้อมูลของระบบไฟล์เหมือนกับ Scandisk บน windows(Faster file system checking) ในระบบไฟล์ Ext3 สามารถเปลี่ยนไปเป็น Ext4 ด้วยวิธีง่ายๆเพียงแค่ 2 คำสั่ง คือ tune2fs และ fsck(Compatibility) แก้ปัญหา Year 2038 problem ด้วยการเพิ่มไปอีก 2 bit ที่ timestamp field ทำให้ขยายเวลาไปอีกกว่า 500 ปี(Improved timestamps) เพิ่มขนาดของ inode เป็น 256 bytes (จากของเดิม 128 bytes ใน Ext3) ไฟล์ใน ลินุกซ์ จะจัดแบ่งและเรียงลำดับกันเป็นโครงสร้างต้นไม้(Tree structure) มีด้วยกันหลายประเภท ดังต่อไปนี้ [14]

- Directory เป็นไฟล์ที่ใช้เก็บรายการของไฟล์อื่นๆ จะขึ้นต้นไฟล์ด้วยอักษร d เช่น

```
drwxr-xr-x 26 root root 4096 Sep 22 09:29 /
```

- Regular files เป็นไฟล์ที่ใช้งานปกติทั่วไป เมื่อทำการสร้างไฟล์ขึ้นบนลินุกซ์ ไฟล์ที่สร้างขึ้นจะเป็นประเภท regular เสมอ

```
-rw-r-—r-- ... /etc/passwd
```

- Special files หรือ Device file เป็นไฟล์ที่ถูกนำมาใช้งานในกรณีพิเศษต่างๆ โดยส่วนมากจะใช้ในการติดต่อกับอุปกรณ์ต่างๆ จะพบเห็นใน /dev

```
crw------- ... /dev/null
brw-rw---- ... /dev/sda
```

- Links เป็นไฟล์ที่ใช้ในการอ้างถึงหรือชี้ไปยังไฟล์ต่างๆ ในระบบ โดยข้อมูลไม่ได้เก็บอยู่ที่ไฟล์นี้

```
lrwxrwxrwx ... termcap -> /usr/share/misc/termcap
```

- Sockets เป็นไฟล์ชนิดพิเศษ เพื่อให้ระบบใช้สำหรับควบคุมงานหรือข้อมูลบางอย่าง

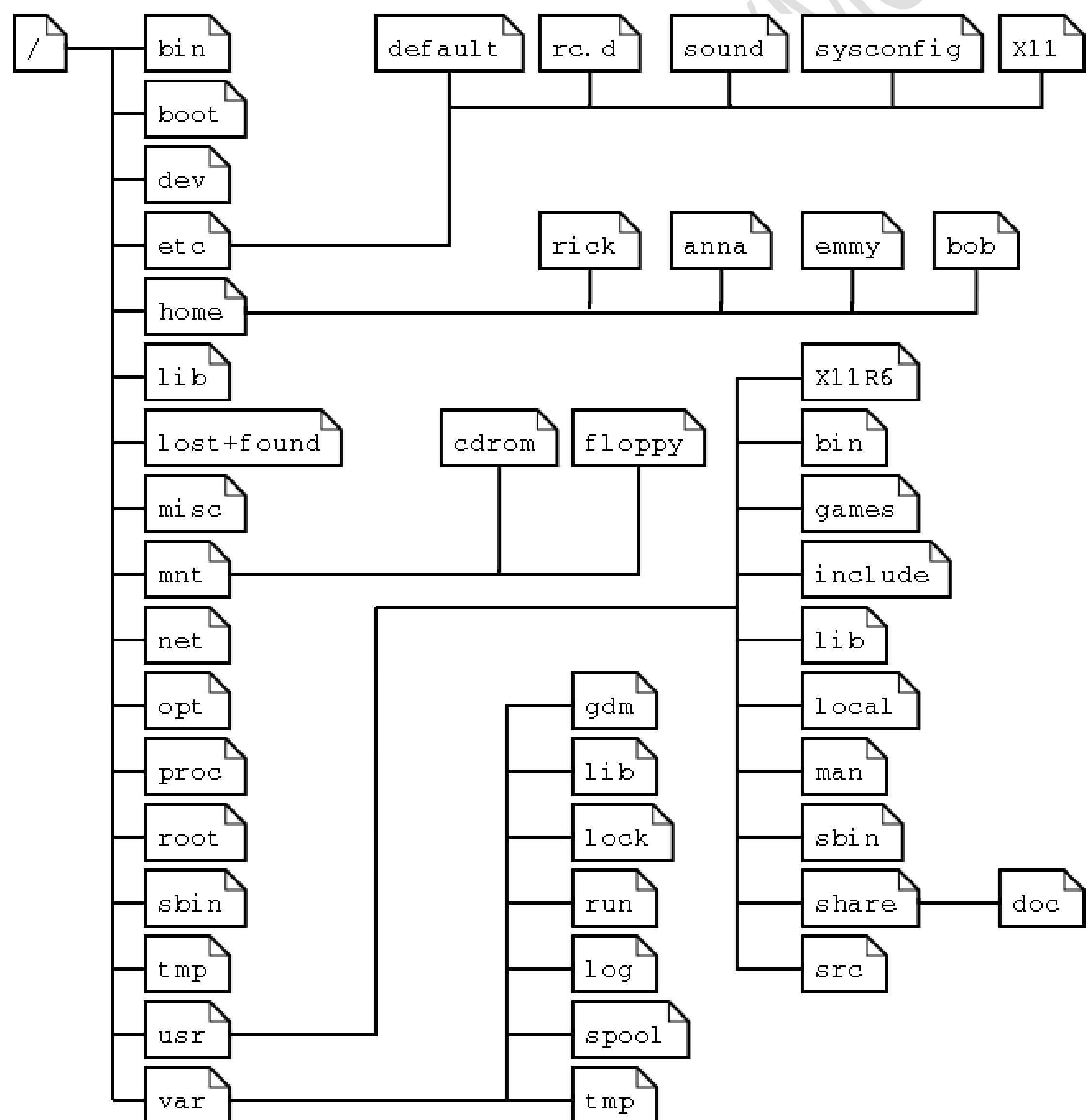
```
srwxrwxrwx ... /tmp/.X11-unix/X0
```

- Named pipes ลักษณะการใช้งานคล้ายกับ socket เพื่อใช้ระหว่างแลกเปลี่ยนข้อมูลระหว่างโปรเซส

```
prw-rw---- ... mypipe
```

โครงสร้างของไดเรคทรอรีของลินุกซ์

ยูนิกซ์และลินุกซ์จะมองทุกอย่างเป็นโครงสร้างของไฟล์ทั้งหมด เช่น CDROM จะอ้างถึงด้วย /media/cdrom ในส่วนนี้จะกล่าวถึงโครงสร้างของไฟล์ต่างๆ พร้อมทั้งหน้าที่การทำงาน



โครงสร้างของไดเรคทรอรีและไฟล์ของลินุกซ์ [8-9]

- / เป็นไดเรคทรอรีเริ่มต้น(root) ซึ่งเป็นไดเรกทอรีที่อยู่ชั้นบนสุดของโครงสร้างลำดับชั้นของไดเรคทรอรี
- /bin ย่อมาจาก Binary เป็นไดเรคทรอรีสำหรับเก็บคำสั่งที่เรียกใช้จากผู้ใช้ โดยส่วนใหญ่เป็นคำสั่งอำนวยความสะดวกแก่ผู้ใช้งานระบบ และเป็นคำสั่งพื้นฐานที่จำเป็นต่อการใช้งานลินุกซ์ ทั่วไป
- /boot ใช้เก็บไฟล์ที่ใช้สำหรับควบคุมการทำงานของลินุกซ์หรือ kernel รวมทั้งไฟล์ที่จำเป็นต้องใช้ boot เครื่องคอมพิวเตอร์ เช่น GRUB (GRand Undified Boot loader)หรือ LILO เป็นต้น
- /dev ย่อมาจาก Device ใช้เก็บชื่อไฟล์พิเศษ ที่เกี่ยวข้องกับอุปกรณ์ต่างๆ ที่มีในระบบ เช่น tty(Terminal) hda(hard disk) เป็นต้น โดยในแต่ละไฟล์จะหมายถึงการเชื่อมต่อกับอุปกรณ์หนึ่งอุปกรณ์
- /etc คำว่า et cetera (ย่อมาจาก Et cetera เป็นภาษาลาติน มีหมายความในภาษาอังกฤษว่า and the others (Et คือ and, cetera คือ others)ใช้เก็บ Configuration file ซึ่งใช้สำหรับดูแลรักษาระบบ (System administrator) และไฟล์สคริปต์ที่ใช้ควบคุมการเปิดปิดบริการ (Service) ต่างๆ ซึ่งส่วนใหญ่จะเก็บอยู่ภายใต้ etc คือ /etc/init.d
- /home เป็นไดเรคทรอรีสำหรับเก็บข้อมูลของผู้ใช้แต่ละคนยกเว้น root จะเก็บไว้ที่ /root แยกออกต่างหาก
- /lib ย่อมาจาก Library ใช้สำหรับเก็บไลบรารี่ของโปรแกรมต่างๆ
- /lost+found เป็นไดเรคทรอรีที่ใช้เก็บไฟล์ที่กำลังถูกใช้งานแล้วดิสก์เกิดปัญหา หรือระบบล้มเหลว หรือไม่ใช้คำสั่งปิดเครื่องที่ถูกต้อง ซึ่งบางครั้งทำให้ไฟล์ที่กำลังใช้งานเหล่านี้มีปัญหา เมื่อเริ่มการทำงานของระบบใหม่(boot) โปรแกรม fsck ซึ่งทำหน้าที่ตรวจสอบระบบไฟล์จะเริ่มทำงาน หากพบไฟล์ที่เกิดความผิดพลาดเกิดขึ้นในระบบไฟล์ใด ก็จะนำไฟล์ที่เก็บสำรองไว้ใน lost+found ไปแทนที่ในไดเรคทรอรีที่มีปัญหาดังกล่าว
- /media เป็นไดเร็กทอรี่สำหรับการเชื่อมอุปกรณ์เข้ากับโครงสร้างของไฟล์ในลินุกซ์ด้วยคำสั่ง mount ส่วนใหญ่อุปกรณ์ที่เชื่อมต่อจะเป็นอุปกรณ์เก็บข้อมูลประเภทสื่อต่างๆ ที่จำเป็นต่อระบบ เช่น cdrom, thumb drive
- /mnt เป็นไดเร็กทอรี่ที่ใช้เชื่อมต่อกับอุปกรณ์เก็บข้อมูลอื่นๆ
- /opt เป็นไดเรคทรอรีที่ใช้เก็บโปรแกรมส่วนขยายอื่นๆ หรือโปรแกรมประเภท third party software หรือโปรแกรมที่เราเอามาติดตั้งเอง แต่ส่วนมากนิยมติดตั้งไว้ใน /usr/local มากกว่า

- /proc เป็นไดเร็กทอรี่สำหรับเก็บข้อมูลของระบบที่กำลังทำงานอยู่ เช่น ข้อมูลโปรเซส หรือสถานะต่างๆ ของระบบ เป็นต้น
- /root เป็น home directory ของ root
- /sbin เป็นไดเรคทรอรีสำหรับเก็บโปรแกรมหรือคำสั่งสำหรับผู้ดูแลระบบ โดยมากโปรแกรมที่เก็บอยู่ในไดเรคทรอรีนี้ต้องใช้ สิทธิ root หรือต้องใช้ sudo จึงจะสามารถใช้งานคำสั่งได้
- /tmp เป็นไดเรคทรอรีที่ใช้เก็บไฟล์ชั่วคราว ไดเรคทรอรีนี้ไม่ว่าผู้ใช้คนใดในระบบสามารถเขียนข้อมูลลงไปได้ และไม่สามารถเก็บข้อมูลได้เมื่อ boot เครื่องใหม่เพราะข้อมูลจะหาย
- /usr เป็นไดเรคทรอรีที่ใช้เก็บโปรแกรม ไลบรารีต่างๆ หรือโปรแกรมต่างๆ ที่ผู้ใช้ติดตั้งเพิ่มเติมลงไป
- /usr/bin เก็บคำสั่งของผู้ใช้งานทั่วไป
- /var เก็บไฟล์ที่เปลี่ยนแปลงหรือไฟล์ชั่วคราวต่างๆ ที่สร้างโดยโปรแกรมหรือผู้ใช้ เช่น log ข้อมูล E-mail ข้อมูลการพิมพ์ต่างๆ

สรุปท้ายบท

โครงสร้างของยูนิกซ์และลินุกซ์ประกอบด้วย 4 ส่วนหลักๆ คือ คือ ฮาร์ดแวร์ คือส่วนประกอบของเครื่องคอมพิวเตอร์ ที่สามารถมองเห็นด้วยตาและสัมผัสได้ เคอร์เนล คือซอฟต์แวร์ที่ทำหน้าที่ เชื่อมต่อและควบคุมการทำงานระหว่างฮาร์ดแวร์กับผู้ใช้งาน เชลล์ คือซอฟต์แวร์ตัวกลางที่ทำหน้าที่ติดต่อระหว่างผู้ใช้กับเคอร์เนล และยูทิลิตี้กับซอฟต์แวร์ประยุกต์ คือซอฟต์แวรที่ทำหน้าที่อำนวยความสะดวกให้กับผู้ใช้งาน

โครงสร้างของไฟล์ระบบปฏิบัติการยูนิกซ์และลินุกซ์มีหลายประเภท คือ FS, Ext FS, Ex2, Ex2 FS, Ex3, Ex4 โดยมีความสามารถเพิ่มขึ้นตามลำดับ โครงสร้างของไฟล์และไดเรคทรอรีของยูนิกซ์และลินุกซ์แบ่งออกเป็นส่วนๆ ตามหน้าที่การทำงาน เช่น /boot มีหน้าที่เก็บแฟ้มสำหรับบูตระบบปฏิบัติการ /bin เก็บคำสั่งที่เรียกใช้จากผู้ใช้งาน /dev เก็บไฟล์ที่เกี่ยวข้องกับอุปกรณ์ต่างๆ ที่มีในระบบ เป็นต้น

คำถามท้ายบท

1. โครงสร้างของยูนิกซ์และลีนุกซ์แบ่งออกได้เป็นกี่ส่วน อะไรบ้าง
2. เคอร์เนล(kernel) ทำหน้าที่อะไรในระบบปฏิบัติการ
3. เชลล์(shell) ทำหน้าที่อะไรในระบบปฏิบัติการ
4. จงอธิบายความแตกต่างของ X-Window, Window Manager, X Desktop
5. จงอธิบายพัฒาการของระบบไฟล์ระบบปฏิบัติการบนลินุกซ์
6. ระบบไฟล์แบบ journal มีคุณสมบัติอย่างไรบ้าง
7. จงอธิบายความแตกต่างของระบบไฟล์แบบ Ext3 และ Ext4
8. จงอธิบายว่าประเภทของไฟล์บนระบบปฏิบัติการลินุกซ์มีกี่ประเภท อะไรบ้าง แต่ละประเภทสามารถจำแนกได้อย่างไร
9. จงอธิบายโครงสร้างไดเรคทอรีและไฟล์บนลินุกซ์ประกอบไปด้วยอะไรบ้าง และแต่ละส่วนทำหน้าที่อะไร เช่น /, /etc, /bin เป็นต้น ให้อธิบายทั้งหมด



บทที่

3

สื่อที่ใช้จัดเก็บระบบปฏิบัติการ

วัตถุประสงค์

1. สามารถอธิบายถึงอุปกรณ์หรือสื่อที่ใช้จัดเก็บระบบปฏิบัติการที่นิยมในปัจจุบันได้
2. สามารถอธิบายถึงโครงสร้างของฮาร์ดดิสก์ว่าประกอบไปด้วยอะไรบ้าง และแต่ส่วนทำหน้าที่อะไร
3. สามารถอธิบายถึงฮาร์ดดิสก์แบบ IDE, SCSI, SATA ได้
4. สามารถอธิบายถึง Hard disk Layout ว่ามีส่วนประกอบอะไรบ้าง
5. สามารถอธิบายถึงความสำคัญของ MBR record
6. สามารถอธิบายถึงการสร้างพาร์ติชันบนฮาร์ดดิสก์

บทที่ 3
สื่อที่ใช้จัดเก็บระบบปฏิบัติการ

บทนำ

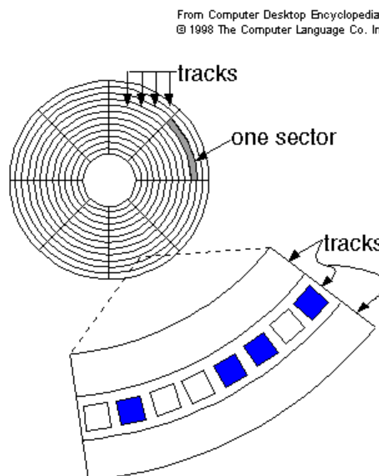
ในบทนี้กล่าวถึงสื่อที่ใช้จัดเก็บระบบปฏิบัติซึ่งมีหลายชนิด เช่น ซีดีรอม ดีวีดี ฮาร์ดดิสก์ เป็นต้น สำหรับสื่อที่นิยมใช้ในปัจจุบันคือฮาร์ดดิสก์เพราะมีความจุสูง ความเร็วในการอ่านและเขียนข้อมูลมาก ราคาถูก ในบทนี้ผู้เขียนเน้นกล่าวถึงสื่อเก็บข้อมูลแบบฮาร์ดดิสก์อย่างละเอียด ฮาร์ดดิกส์ที่ใช้ในปัจจุบันมี 3 ประเภทหลักคือ IDE, SATA และ SCSI ในปัจจุบันได้มีการพัฒนาฮาร์ดดิสก์แบบโซริสเตจ เพิ่มขึ้นมาเพื่อเพิ่มขีดความสามารถในการอ่านเขียนข้อมูล

อุปกรณ์หรือสื่อที่ใช้จัดเก็บระบบปฏิบัติการที่นิยมในปัจจุบันซึ่งมี 2 ประเภทคือ CD/DVD และฮาร์ดดิสก์

ซีดีรอม (CD-ROM หรือ Compact Disk Read Only Memory) [17]

แผ่นซีดีรอมมี 2 ประเภทใหญ่ๆ คือ CD-RW(เขียนข้อมูลได้หลายครั้ง) และ CD-R(เขียนข้อมูลได้ครั้งเดียว) สามารถเก็บข้อมูลได้สูงถึง 650 เมกะไบต์ต่อแผ่น การใช้งานแผ่นซีดีรอมจะต้องมีเครื่องคอมพิวเตอร์ที่มีซีดีรอมไดร์ฟ (CD-ROM Drive) ซีดีรอมได้รับความนิยมใช้เป็นสื่อเก็บข้อมูลเป็นอย่างมากเนื่องจากมีความจุสูง เช่น ซอฟต์แวร์เกมส์ เพลง หนังสือ ภาพยนตร์ รูปภาพมัลติมีเดีย เป็นต้น และปัจจุบันนิยมติดตั้งระบบปฏิบัติการไว้บนซีดีรอมเป็นแบบ Live-CD





ซีดีรอมและโครงสร้างของซีดีรอม

รูปที่ 3-1 ซีดีรอม

ดีวีดี (DVD หรือ Digital Versatile Disk)

ดีวีดี เป็นเทคโนโลยีใหม่ล่าสุดที่มีแนวโน้มจะได้รับความนิยมสูงสุด โดยแผ่นดีวีดีสามารถเก็บข้อมูลได้ต่ำสุดที่ 4.7 จิกะไบต์ ซึ่งเพียงพอสำหรับเก็บภาพยนตร์เต็มเรื่องด้วยคุณภาพระดับ

สูงสุดทั้งภาพและเสียง (ในขณะที่ CD-ROM หรือ Laser Disk ที่นิยมใช้เก็บภาพยนต์ในปัจจุบันต้องใช้หลายแผ่น) ทำให้เป็นที่คาดหมายว่าดีวีดีจะมาแทนที่ทั้งซีดีรอม เลเซอร์ดสิก์หรือแม้กระทั้งวีดีโอเทป ข้อกำหนดของดีวีดีจะสามารถมีความจุได้ตั้งแต่ 4.7 GM ถึง 17 GM และมีความเร็วในการเข้าถึง (Access time) อยู่ที่ 600 กิโลไบต์ต่อวินาที ถึง 1.3 เมกะไบต์ต่อวินาที รวมทั้งสามารถอ่านแผ่นซีดีรอมแบบเก่าได้ด้วย

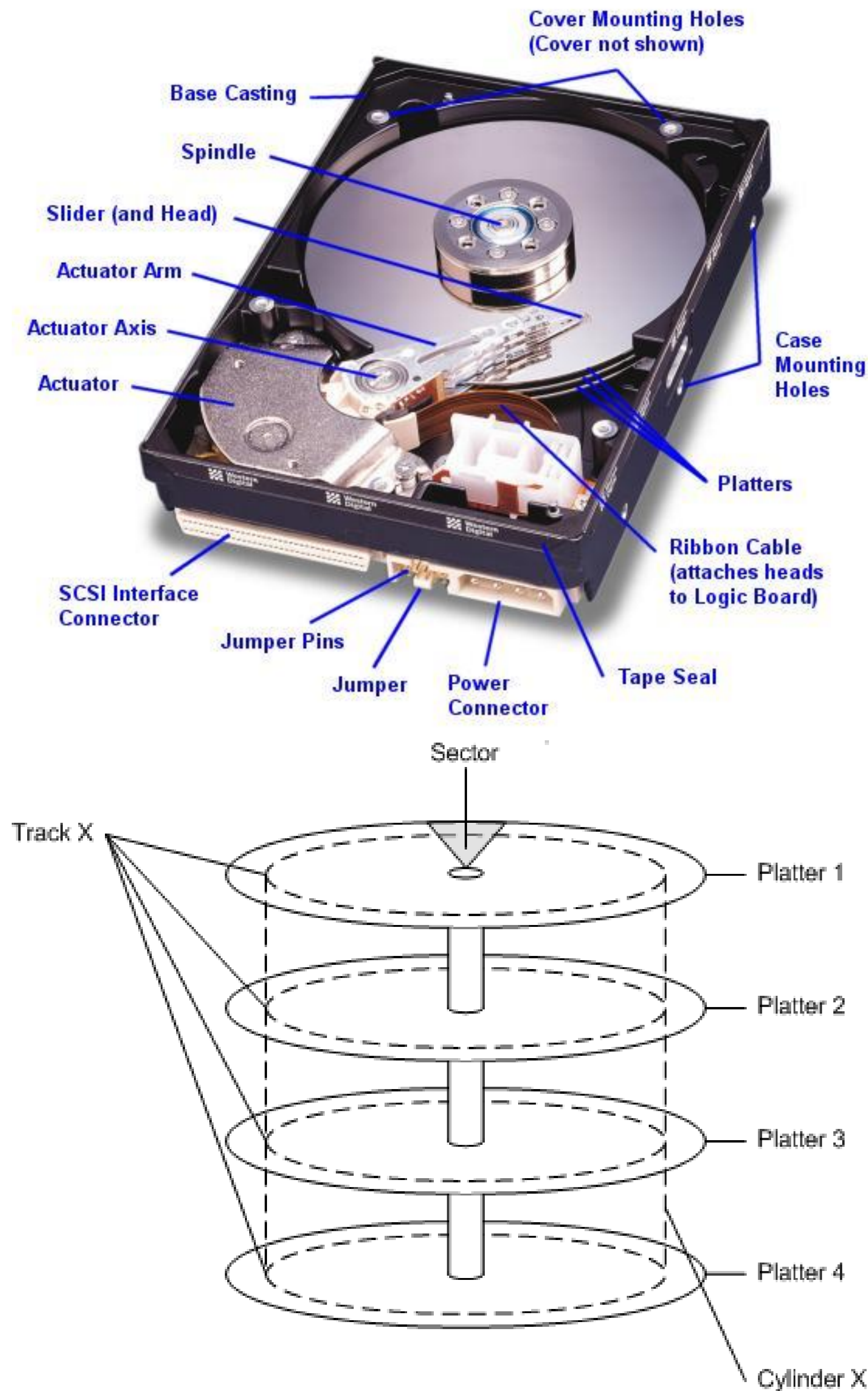


แผ่นดีวีดีและดีวีดีไดร์ฟ

รูปที่ 3-2 ดีวีดี

ฮาร์ดดิสก์ (Hard Disk)

ฮาร์ดดิสก์ ประกอบไปด้วยจานแม่เหล็กหรือจานดิสก์ (Platter) ซึ่งออกแบบมาสำหรับบันทึกข้อมูลโดยขึ้นอยู่กับสถาปัตยกรรมในการออกแบบด้วยว่าได้มีการกำหนดให้มีขนาดความจุต่อแผ่นเท่าใด และในฮาร์ดดิสก์ แต่ละรุ่นจำต้องใช้จำนวนแผ่นเท่าใด ซึ่งจานแม่เหล็กมีลักษณะเป็นทรงกลมและมีมอเตอร์สำหรับควบคุมการหมุนของจานดิสก์ (Spindle)โดยอัตราความเร็วในการหมุน ปัจจุบันถูกจัดกลุ่มออกเป็น 5400, 7200 และ 10,000 รอบต่อนาที(rpm) ซึ่งถ้าจำนวนรอบในการหมุนของจานดิสก์มีระดับความถี่ที่สูง ก็จะส่งผลให้สามารถเข้าถึงข้อมูลได้รวดเร็วยิ่งขึ้นตามไปด้วย การบันทึกข้อมูลในฮาร์ดดิสก์จะแบ่งเป็นวงรอบเรียกว่า แทร็ก (track) ฮาร์ดดิสก์จะเก็บข้อมูลเป็นวงครบรอบหลาย ๆ วง ฮาร์ดดิสก์ต้องทำการจัดรูปแบบแผ่นก่อน หรือที่เรียกว่า การฟอร์แมต (format) ขั้นตอนการฟอร์แมตเริ่มจากการสร้างแทร็กก่อน และในแต่ละแทร็กจะแบ่งออกเป็นส่วน ๆ ที่เรียกว่า เซ็กเตอร์ (sector) ความจุของฮาร์ดดิสก์สามารถคำนวณจากจำนวนแผ่นบันทึกข้อมูล จำนวนแทร็กในแต่ละแผ่นคูณกับจำนวนเซ็กเตอร์ในแต่ละแทร็ก โดยหนึ่งเซ็กเตอร์จะมีเนื้อที่เก็บข้อมูลเท่ากับ 512 ไบต์ [18]



รูปที่ 3-3 โครงสร้างฮาร์ดดิสก์

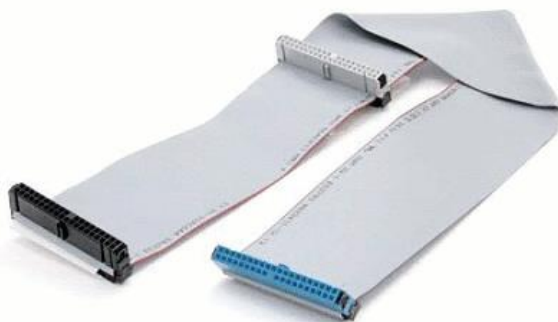
อินเตอร์เฟสของฮาร์ดดิสก์ที่ใช้ในปัจจุบัน มีอยู่ 3 ชนิดด้วยกัน [19] คือ

IDE (Integrated Drive Electronics)

เป็นอินเตอร์เฟสที่นิยมใช้กันมาก แต่ในปัจจุบันเริ่มจะไม่นิยมใช้งานแล้ว เนื่องจากอัตราการโอนถ่ายข้อมูลต่ำกว่าประเภทอื่นๆ การต่อไดร์ฟฮาร์ดดิสก์แบบ IDE จะต่อผ่านสายแพรและ

คอนเน็คเตอร์จำนวน 40 ขาที่มีอยู่บนเมนบอร์ด ส่วนใหญ่แล้วใน 1 คอนเน็คเตอร์ จะสามารถต่อฮาร์ดดิสก์ได้ 2 ตัว ในคอมพิวเตอร์ในปัจจุบันจะสามารถต่อฮาร์ดดิสก์ชนิด IDE ได้ไม่เกิน 4 ตัว ฮาร์ดดิสก์แบบ IDE มีความเร็วสูงสุดประมาณ 8.3 Mbytes ต่อวินาที และ EIDE(Enchanced IDE) มีความเร็วสูงสุดประมาณ 133.3 Mbytes ต่อวินาที





Hard disk IDE IDE Cable

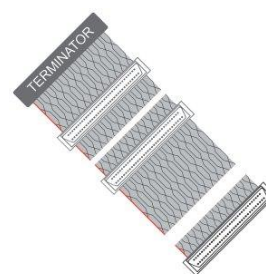
รูปที่ 3-4 ฮาร์ดดิสก์ชนิด IDE

SCSI (Small Computer System Interface)

เป็นอินเตอร์เฟสที่แตกต่างจากอินเตอร์เฟสแบบอื่น ๆ มาก โดยจะอาศัย Controller Card ที่มี Processor อยู่ในตัวเอง เมื่อต้องการใช้งานจำเป็นต้องต่อเพิ่มการ์ด Controller เข้ากับเมนบอร์ดของเครื่องคอมพิวเตอร์ อินเตอร์เฟสประเภทนี้สนับสนุนการเชื่อมต่อฮาร์ดดิสก์ได้สูงถึง 8 ตัว แต่การ์ดบางรุ่นอาจจะได้ถึง 14 ตัวเลยทีเดียว โดยส่วนใหญ่แล้วจะใช้งานกับเครื่องเซิร์ฟเวอร์ เพราะอินเทอร์เฟสประเภทนี้มีราคาแพง แต่ให้ความเร็วในการโอนถ่ายข้อมูลสูงถึง 320 Mbytes ต่อวินาที







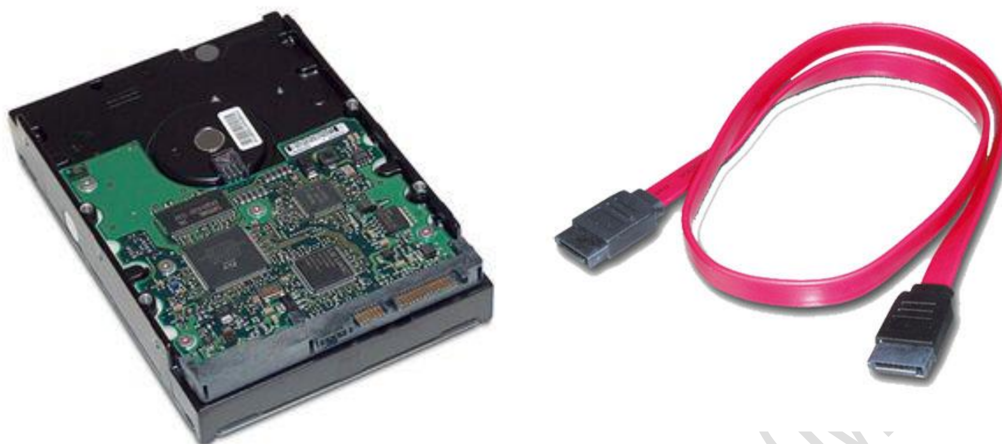
Harddisk SCSI SCSI Controller SCSI Cable

รูปที่ 3-5 ฮาร์ดดิสก์ชนิด SCSI

Serial ATA (Advanced Technology Attachment)

เป็นอินเตอร์เฟสแบบใหม่ เปิดตัวครั้งแรกในวันที่ 26 มิถุนายน 2545 งาน PC Expo ใน New York มีความเร็วในเข้าถึงข้อมูลถึง 150 Mbytes ต่อ วินาที ปัจจุบัน SATA 2 มีความเร็วสูงถึง

300 Mbytes ต่อวินาที และให้ผลตอบสนองในการทำงานได้เร็วมากในส่วนของ extreme application เช่น Game Home Video และ Home Network Hub โดยเป็นอินเตอร์เฟสที่จะมาแทนที่ของ IDE ในปัจจุบัน



Hard disk Serial ATA(SATA) SATA Cable

รูปที่ 3-6 ฮาร์ดดิสก์ชนิด SATA

Hard disk Layout [20]

ฮาร์ดดิสก์สามารถแบ่งออกเป็นส่วนๆ ได้ดังต่อไปนี้

- MBR (Master Boot Record)
- Boot Sector
- Primary Partition
- Extend Partition
- Logical Partition

MBR เป็นส่วนที่ใช้กำหนดตำแหน่งการ boot ของระบบปฏิบัติการบนฮาร์ดดิสก์ พื้นที่ในส่วนนี้จะอยู่ sector แรกบนจานแม่เหล็ก(sector 0) มีขนาดไม่เกิน 512 ไบต์ พื้นที่ของ MBR ไม่ใช่พื้นที่ที่สำหรับใช้งานทั่วๆ ไป แต่จะมีหน้าที่อย่างใดอย่างหนึ่งหรือทั้งหมดดังนี้

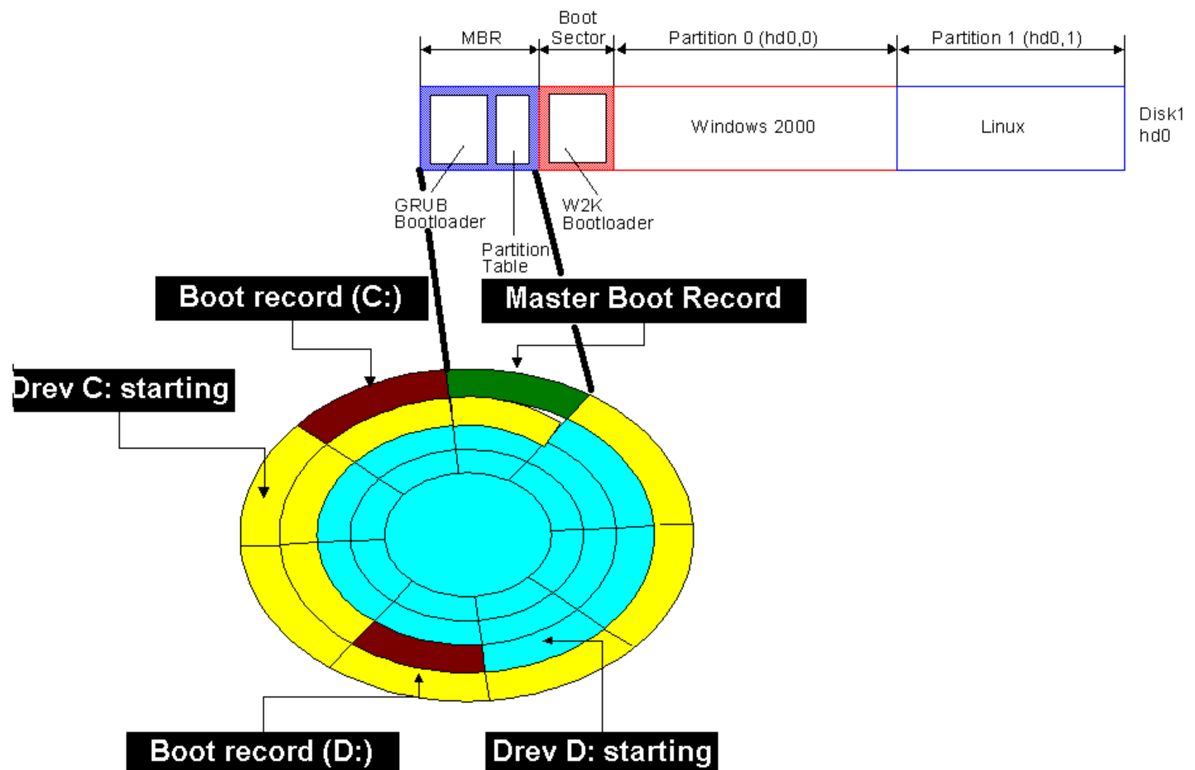
- เก็บรหัสคำสั่งเพื่อชี้ไปยัง Primary Partition
- เก็บรหัสคำสั่งที่ทำงานในส่วนของ Bootstrapping operating systems หลังจากที่ Bios ได้ผ่านการตรวจสอบฮาร์ดแวร์เสร็จเรียบร้อยแล้ว

MBR จะแบ่งออกเป็น 3 ส่วนคือ

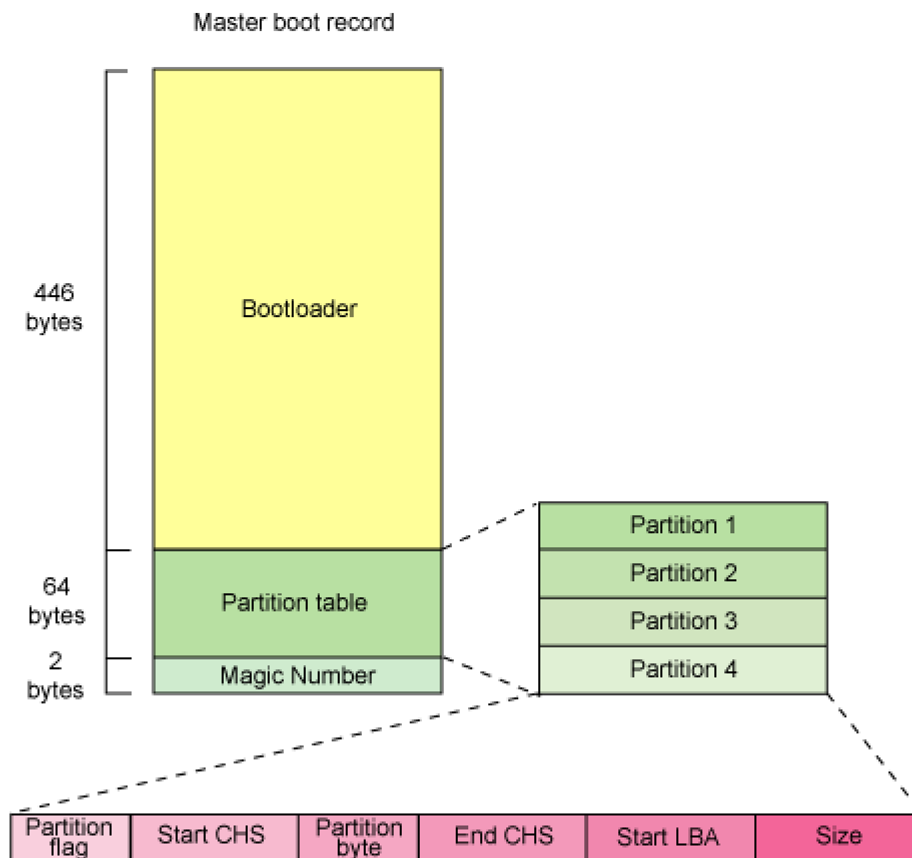
IPL(Initial Program Loader) มีขนาด 446 ไบต์ ทำหน้าที่เก็บโปรแกรม Boot Loader เป็นรหัสคำสั่งที่ทำหน้าที่โหลดระบบปฏิบัติการที่อยู่บนฮาร์ดดิสก์ขึ้นมาทำงานต่อจาก Bios ได้

Partition Table มีขนาด 66 ไบต์ ทำหน้าที่เก็บรหัสคำสั่งที่ชี้ไปยัง Partition ที่เก็บระบบปฏิบัติการอยู่

Magic Number มีขนาด 2 ไบต์



รูปที่ 3-7 โครงสร้างของ Disk Layout



รูปที่ 3-8 Anatomy of the MBR [21]

Address			Description		Size in bytes
Hex	Oct	Dec			
0000	0000	0	Code Area		**440** (max. 446)
01B8	0670	440	Optional Disk signature		4
01BC	0674	444	Usually Nulls; 0x0000		2
01BE	0676	446	**Table of primary partitions** (Four 16-byte entries, IBM Partition Table scheme)		**64**
01FE	0776	510	55h	MBR signature; 0xAA55[1]	**2**
01FF	0777	511	AAh		
MBR, total size: 446 + 64 + 2 =					**512**

รูปที่ 3-9 Structure of a Master Boot Record

การสร้าง Partition บนฮาร์ดดิสก์ สามารถทำได้ 3 แบบคือ Primary, Extended และ Logical สำหรับ Primary Partition มีข้อจำกัดว่าสามารถสร้าง ได้เพียง 4 Partition เท่านั้น ถ้าต้องการแบ่ง Partition ให้ได้มากกว่านั้น ต้องอาศัย Partition แบบ Extended ช่วย คือเมื่อทำการสร้าง Partition แบบ Extended แล้วสามารถสร้าง Partition ย่อยๆ เรียกว่า Logical ได้ถึง 63 พาร์ติชันเลยทีเดียว ดังตัวอย่างการแบ่ง partition ต่อไปนี้

MBR	1 PRIMARY	2 PRIMARY	3 PRIMARY	4 PRIMARY

รูปที่ 3-10 การแบ่ง Partition แบบ Primary ทั้งหมด 4 พาร์ติชัน

MBR	1 PRIMARY	2 PRIMARY	3 PRIMARY	4 Extended	5 Logical	6 Logical	... Logical	n Logical

รูปที่ 3-11 การแบ่ง Partition โดยใช้พาร์ติชันที่ 4 เป็น Extended

MBR	1 PRIMARY	2 PRIMARY	3 Extended	5 Logical	6 Logical	... Logical	n Logical

รูปที่ 3-12 การแบ่ง Partition โดยใช้พาร์ติชันที่ 3 เป็น Extended

MBR	1 PRIMARY	2 Extended	5 Logical	6 Logical	... Logical	n Logical

รูปที่ 3-12 การแบ่ง Partition โดยใช้พาร์ติชันที่ 2 เป็น Extended

การแบ่งพาร์ติชันบนลินุกซ์ มีข้อจำกัดคือ พาร์ติชันแรกที่เป็น Logical จะเป็นพาร์ติชันที่ 5 เสมอ และฮาร์ดดิสก์แบบ IDE สามารถแบ่งพาร์ติชันได้สูงที่สุดไม่เกิน 63 พาร์ติชัน สำหรับฮาร์ดดิสก์แบบ SCSI ได้สูงสุดไม่เกิน 15 พาร์ติชัน

การเรียกชื่อฮาร์ดดิสก์และพาร์ติชัน [22]

ลินุกซ์จะมองเห็นอุปกรณ์ทุกอย่างเป็นไฟล์ เรียกว่า device file อยู่ภายใต้ directory ชื่อ /dev ฮาร์ดดิสก์ก็จะถูกมองเป็น device file เช่นกัน วิธีการตั้งชื่อ device file ของฮาร์ดดิสก์ จะขึ้นอยู่กับประเภท ช่องสัญญาณ IDE ที่เชื่อมต่อและลำดับของพาร์ติชั่นของฮาร์ดดิสก์ตัวนั้น ดังนี้คือ

ตัวอักษร 2 ตัวแรก หมายถึง ประเภทของฮาร์ดดิสก์ ถ้าเป็นแบบ IDE จะมีชื่อเป็น hd ถ้าเป็น SCSI จะแทนด้วยตัวอักษร sd

ตัวอักษรตัวที่ 3 หมายถึง ตำแหน่งของช่องสัญญาณ IDE ที่ต่อกับฮาร์ดดิสก์ มีอยู่ 4 ตัวดังนี้

a หมายถึง ฮาร์ดดิสก์ตัวที่ 1 หรือ primary master (/dev/hda)

b หมายถึง ฮาร์ดดิสก์ตัวที่ 2 หรือ primary slave (/dev/hdb)

c หมายถึง ฮาร์ดดิสก์ตัวที่ 3 หรือ secondary master (/dev/hdc)

d หมายถึง ฮาร์ดดิสก์ตัวที่ 4 หรือ secondary slave (/dev/hdd)

ตัวเลขตัวสุดท้าย หมายถึง หมายเลขพาร์ติชั่นของฮาร์ดดิสก์ ตัวอย่างคือ

/dev/hda1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 1 เป็น primary master

/dev/hda2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 1 เป็น primary master

/dev/hda3 หมายถึง พาร์ติชั่นที่ 3 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 1 เป็น primary master

/dev/hdan หมายถึง พาร์ติชั่นที่ n ของฮาร์ดดิสก์แบบ IDE ตัวที่ 1 เป็น primary master

/dev/hdb1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 2 เป็น primary slave

/dev/hdb2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 2 เป็น primary slave

/dev/hdbn หมายถึง พาร์ติชั่นที่ n ของฮาร์ดดิสก์แบบ IDE ตัวที่ 2 เป็น primary slave

/dev/hdc1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 3 เป็น secondary master

/dev/hdc2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 3 เป็น secondary master

/dev/hdcn หมายถึง พาร์ติชั่นที่ n ของฮาร์ดดิสก์แบบ IDE ตัวที่ 3 เป็น secondary master

/dev/hdd1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 4 เป็น secondary slave

/dev/hdd2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ IDE ตัวที่ 4 เป็น secondary slave

/dev/hddn หมายถึง พาร์ติชั่นที่ n ของฮาร์ดดิสก์แบบ IDE ตัวที่ 4 เป็น secondary slave

สำหรับการเรียกชื่อฮาร์ดดิสก์แบบ SCSI จะเรียกตาม SCSI ID เช่น

SCSI ID 0 เรียกชื่อว่า /dev/sda

SCSI ID 1 เรียกชื่อว่า /dev/sdb

SCSI ID 2 เรียกชื่อว่า /dev/sdc

SCSI ID 3 เรียกชื่อว่า /dev/sdd

SCSI ID 4 เรียกชื่อว่า /dev/sd…

ชื่อจะเรียงต่อกันไปเรื่อยๆ (sd…) ตามจำนวนของ SCSI ที่สามารถใส่เข้าไปได้ และสำหรับการอ้างพาร์ติชันก็เรียกชื่อเหมือนกับ IDE คือ /dev/sda1, /dev/sda2,.. , /dev/sdb1, /dev/sdb2,.. , /dev/sdc1, /dev/sdc2 ดังตัวอย่าง

/dev/sda1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ SCSI ID 0

/dev/sda2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ SCSI ID 0

/dev/sdb1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ SCSI ID 1

/dev/sdb2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ SCSI ID 1

/dev/sdc1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ SCSI ID 2

/dev/sdc2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ SCSI ID 2

…

/dev/sdn1 หมายถึง พาร์ติชั่นที่ 1 ของฮาร์ดดิสก์แบบ SCSI ID n

/dev/sdn2 หมายถึง พาร์ติชั่นที่ 2 ของฮาร์ดดิสก์แบบ SCSI ID n

สำหรับฮาร์ดดิสก์แบบ SATA จะเรียกชื่อเหมือนกับฮาร์ดดิสก์แบบ SCSI

การติดตั้งลินุกซ์ลงบนฮาร์ดดิสก์ต้องการอย่างน้อย 2 พาร์ติชั่น คือ root partition และ swap partition ลินุกซ์สามารถติดตั้งได้ทั้งใน primary partition หรือ extended partition ก็ได้

สรุปท้ายบท

ระบบปฏิบัติการยูนิกซ์และลินุกซ์สามารถติดตั้งได้บนสื่อจัดเก็บได้หลายประเภท เมื่อต้องการระบบบฏิบัติการที่สามารถเคลื่อนย้ายได้สะดวกให้เลือกใช้ ซีดี หรือ ดีวีดี แต่ถ้าต้องการติดตั้งบนสื่อจัดเก็บที่ให้ความเร็ว และความจุสูง และไม่ควรเคลื่อนที่ ให้เลือกใช้ฮาร์ดดิสก์ ซึ่งมีให้เลือกหลายประเภท เช่น IDE, SATA, SCSI เป็นต้น ฮาร์ดดิสก์มีส่วนสำคัญที่เรียกว่า MBR ทำหน้าที่กำหนดตำแหน่งการ boot ของระบบปฏิบัติการบนฮาร์ดดิสก์ พื้นที่ในส่วนนี้จะอยู่ sector แรกบนจานแม่เหล็ก การสร้าง Partition บนฮาร์ดดิสก์ สามารถทำได้ 3 แบบคือ Primary, Extended และ Logical

คำถามท้ายบท

1. อุปกรณ์หรือสื่อที่ใช้จัดเก็บระบบปฏิบัติการที่นิยมในปัจจุบันมีกี่ประเภทอะไรบ้าง
2. CD และ DVD แตกต่างกันอย่างไรบ้าง
3. ฮาร์ดดิสก์มีกี่ประเภทอะไรบ้าง
4. จงอธิบายโครงสร้างของฮาร์ดดิสก์ว่าประกอบไปด้วยอะไรบ้าง เช่น Platter, Track, Sector เป็นต้น
5. ฮาร์ดดิสก์แบบ IDE มีคุณสมบัติอย่างไร เช่น ความเร็ว การเชื่อมต่อ เป็นต้น พร้อมอธิบาย
6. ฮาร์ดดิสก์แบบ SCSI มีคุณสมบัติอย่างไร เช่น ความเร็ว การเชื่อมต่อ เป็นต้น พร้อมอธิบาย
7. ฮาร์ดดิสก์แบบ SATA มีคุณสมบัติอย่างไร เช่น ความเร็ว การเชื่อมต่อ เป็นต้น พร้อมอธิบาย
8. Hard disk Layout มีส่วนประกอบอะไรบ้าง อธิบายมาพอเข้าใจ
9. จงอธิบายความสำคัญของ MBR record
10. การสร้างพาร์ติชันบนฮาร์ดดิสก์มีกี่ชนิด อะไรบ้าง
11. จงอธิบายข้อจำกัดของการเชื่อมต่อแบบ IDE, SCSI
12. จงอธิบายการแบ่งพาร์ติชันว่าสามารถทำได้แบบใดบ้าง
13. จงอธิบายถึงการเรียกชื่อของฮาร์ดดิสก์และพาร์ติชันแบบ IDE
14. จงอธิบายถึงการเรียกชื่อของฮาร์ดดิสก์และพาร์ติชันแบบ SCSI
15. จงอธิบายถึงการเรียกชื่อของฮาร์ดดิสก์และพาร์ติชันแบบ SATA



บทที่ 4

กระบวนการบูตและการชัตดาวน์ของลินุกซ์

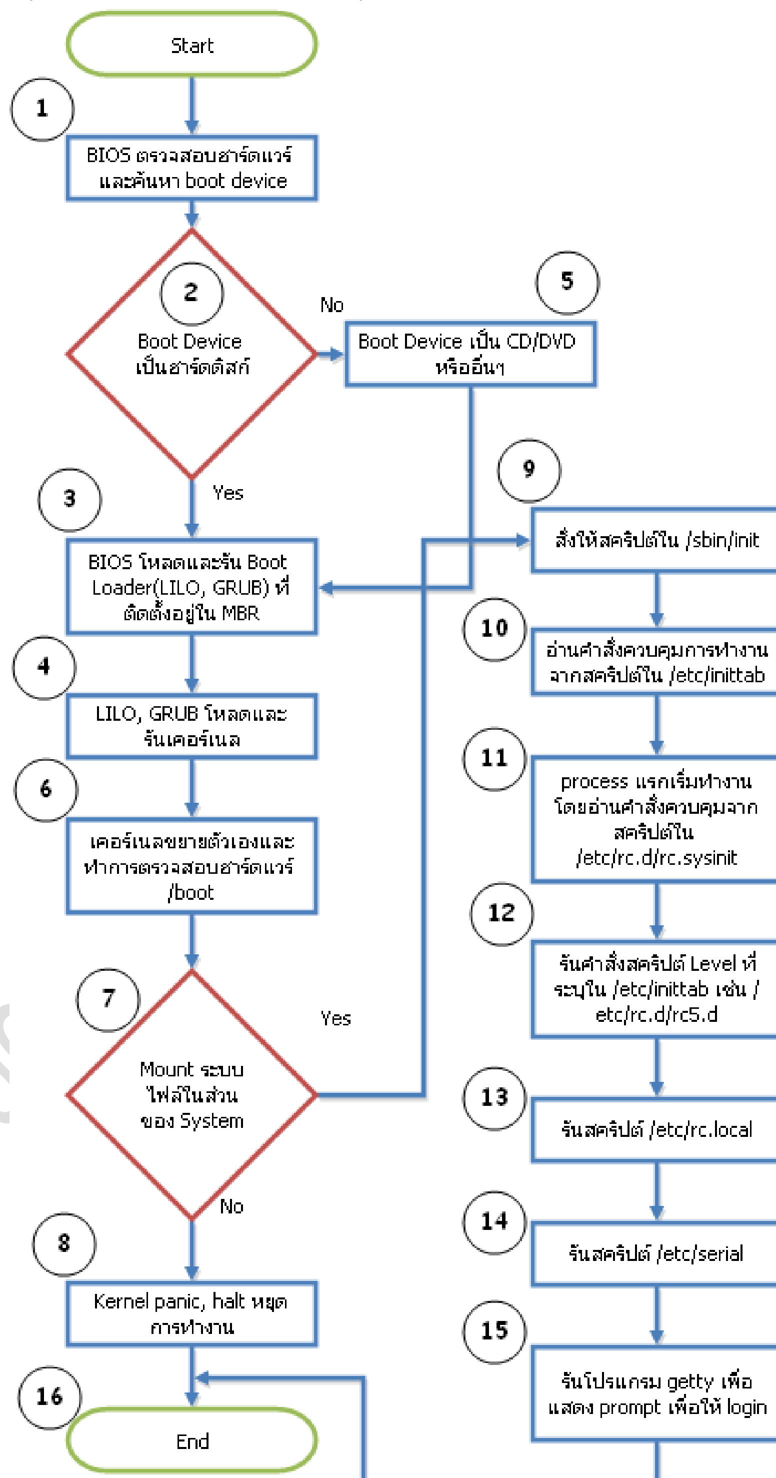
วัตถุประสงค์

1. สามารถอธิบายกระบวนการบูตและชัตดาวน์ระบบปฏิบัติการลินุกซ์ได้อย่างถูกต้อง
2. สามารถอธิบายถึงการทำงานของ BIOS, MBR, LILO และ GRUB คืออะไร ว่ามีหน้าที่และความสำคัญอย่างไร
3. สามารถอธิบายถึงขบวนการ mount file system
4. สามารถอธิบายถึงหน้าที่ของสคริปต์ /sbin/init, /etc/inittab, /etc/rc.d/rc.sysint
5. สามารถอธิบายถึงไดเรคทอรี /etc/rc.d/rc1.d, /etc/rc.d/rc2.d, /etc/rc.d/rc3.d, /etc/rc.d/rc4.d, /etc/rc.d/rc5.d, /etc/rc.local ทำหน้าที่อะไร
6. สามารถอธิบายว่าวิธีการ shutdown/restart ลินุกซ์ สามารถทำได้กี่วิธี อะไรบ้าง

บทที่ 4

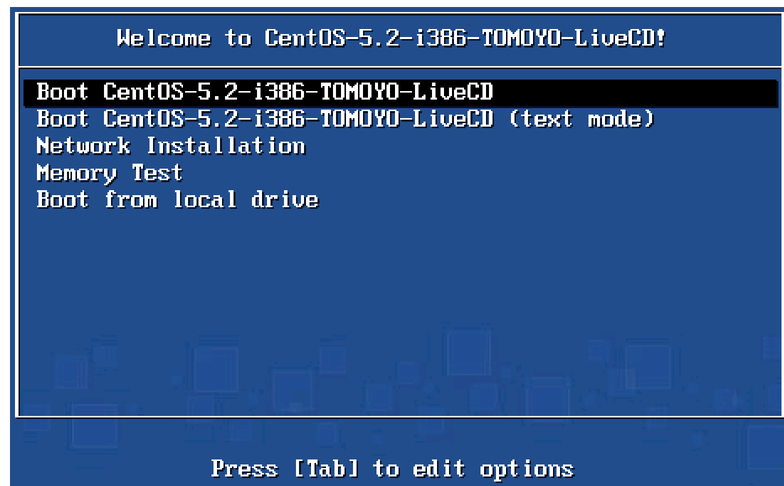
กระบวนการบูตและการชัตดาวน์ของลินุกซ์

ในบทนี้อธิบายถึงขั้นตอนการบูตระบบปฏิบัติการลินุกซ์ โดยมีวัตถุประสงค์เพื่อให้เข้าใจถึงกระบวนการบูตของลินุกซ์ทั้งหมดได้อย่างถูกต้อง [21,22-23] ซึ่งมีขั้นตอนดังต่อไปนี้



รูปที่ 4-1 แสดงกระบวนการบูตของลินุกซ์

1. เริ่มตั้งแต่เปิดเครื่องคอมพิวเตอร์หรือเริ่มต้นคอมพิวเตอร์ใหม่(restart) หน่วยประมวลผลกลาง (processor) จะทำการโหลดและรันรหัสคำสั่งที่บรรจุอยู่ใน BIOS(Basic Input/Output System) ซึ่งเก็บไว้ใน ROM (Read Only Memory) ขึ้นมาทำงาน BIOS จะทำการค้นหาอุปกรณ์ที่เชื่อมต่อพร้อมตรวจสอบ อุปกรณ์ต่อเชื่อมเหล่านั้น เช่น เมาส์ คีย์บอร์ด จอภาพ หน่วยความจำ เป็นต้น ซึ่งกระบวนการในส่วนนี้เรียกว่า power-on self test (POST)
2. ขั้นตอนต่อไปรหัสคำสั่งใน BIOS จะทำการค้นหาอุปกรณ์ที่มีหน้าที่ในการบูตระบบ โดยปกติแล้ว จะค้นหาฟลอปปี้ดิสก์ ซีดีรอม ฮาร์ดดิสก์ ตามลำดับ(เราสามารถปรับลำดับการเลือกอุปกรณ์ที่ใช้บูตได้โดยการ setup ใน BIOS) เมื่ออุปกรณ์ที่ใช้ในการบูตเป็นฮาร์ดดิสก์ก็จะเข้าไปโหลด boot loader ในฮาร์ดดิสก์ แต่ถ้าไม่ใช่ เช่น บูตจาก CD/DVD เป็นต้น ก็จะต้องทำตามระเบียบวิธีการของแต่ละวิธี แต่สุดท้ายก็จะเหมือนกันคือจะต้องโหลดและรัน boot loader เช่นเดียวกัน
3. เซกเตอร์แรกของฮาร์ดดิสก์ จะบรรจุข้อมูลส่วนที่สำคัญที่สุด เรียกว่า Master Boot Record (MBR) โดยใน MBR นี้จะมีโปรแกรมของ boot loader บรรจุไว้อยู่ซึ่งจะมีข้อมูลรายละเอียดของพาร์ทิชัน รวมทั้งโค้ดในการส่งผ่านการทำงานให้เคอร์เนล (ในแต่ละพาร์ทิชันจะมี boot sector แยกเป็นของตัวเองต่างหาก ตัว boot loader เป็นเพียงตัวทำหน้าที่แสดงผลให้ผู้ใช้เลือกและส่งต่อกระบวนการบูตไปให้กับโปรแกรมในส่วน boot sector) boot loader ในลินุกซ์ที่นิยมมี 2 โปรแกรมคือ LILO และ GRUB ปัจจุบันนิยมใช้ GRUB มากกว่า
4. BIOS จะทำการโหลด LILO/GRUB จาก MBR และรัน LILO/GRUB ขึ้นมาพร้อมกับส่งต่อกระบวนการบูตให้แก่ LILO/GRUB โดย LILO/GRUB จะแสดง prompt ให้ผู้ใช้สามารถเลือกได้ว่าจะบูตจากระบบปฏิบัติการใด (ขึ้นกับการตั้งค่าในไฟล์ /etc/lilo.conf, /etc/grub.conf) ไฟล์คอนฟิกดังกล่าวจะมีข้อมูลอยู่แล้วว่าระบบปฏิบัติการใดถูกติดตั้งบนพาร์ติชันใด หากผู้ใช้ไม่ได้เลือกว่าจะบูตจากระบบปฏิบัติการใดหรือเคอร์เนลเวอร์ชันใดภาย ในเวลาที่ระบุไว้ ตัว LILO/GRUB จะทำการบูตจากเคอร์เนลที่เป็นดีฟอลต์โดยอัตโนมัติ



รูปที่ 4-2 ตัวอย่าง Boot Loader(GRUB)

5. กรณีที่บูตจาก CD/DVD ก็จะอ่านข้อมูลของ MBR จากแผ่น CD/DVD แทน
6. หลังจากที่ LILO/GRUB เลือกแล้วว่าจะทำการบูตจากพาร์ทิชันใด มันจะทำการโหลดเคอร์เนลขึ้นสู่หน่วยความจำและเคอร์เนลจะทำหน้าที่ควบคุม กระบวนการทำงานของคอมพิวเตอร์ต่อไป ซึ่งสามารถอธิบายการทำงานได้ดังนี้
    - เคอร์เนลของลินุกซ์จะทำการขยายตัวมันเองก่อนซึ่งบีบอัดไว้ (โดยเคอร์เนลมีโปรแกรมขนาดเล็กที่อยู่ส่วนต้นทำหน้าที่ในการขยายตัวเอง) เนื่องจากไม่ต้องการให้เคอร์เนลมีขนาดใหญ่เกินไป
    - ลินุกซ์จะถามว่าจะให้ใช้การแสดงผลในโหมดใด ระหว่างกราฟฟิกโหมดหรือ Text โหมด
    - เคอร์เนลจะทำการตรวจสอบฮาร์ดแวร์ต่างๆ เช่น ฮาร์ดดิสก์ ฟลอปปี้ การ์ดเครือข่าย รวมทั้ง device driver ที่มันรู้จักว่าทำงานได้ตามปกติหรือไม่ โดยจะมีการแสดงผลออกมาที่คอนโซลในระหว่างกระบวนการนี้ด้วย

```
Loading linux.......................
Uncompressing Linux... Ok, booting the kernel.
Linux version 2.4.7-10 (bhcompile@stripples.devel.redhat.com) (gcc version 2.96
20000731 (Red Hat Linux 7.1 2.96-98)) #1 Thu Sep 6 17:27:27 EDT 2001
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000e7000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000008000000 (usable)
 BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000fffe0000 - 0000000100000000 (reserved)
Scanning bios EBDA for MXT signature
On node 0 totalpages: 32768
zone(0): 4096 pages.
zone(1): 28672 pages.
zone(2): 0 pages.
Kernel command line: auto BOOT_IMAGE=linux ro root=801 BOOT_FILE=/boot/vmlinuz-2
.4.7-10
Initializing CPU#0
Detected 1496.139 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... _
```

รูปที่ 4-3 เริ่มต้นกระบวนการบูตเคอร์เนลของลินุกซ์

- จากนั้นเคอร์เนลจะ mount root filesystem แบบอ่านอย่างเดียว (read only) เพื่อตรวจสอบ file system หากพบข้อผิดพลาดที่ทำให้ไม่สามารถ mount root filesystem ได้ ระบบจะไม่สามารถทำงานได้และจะแฮงค์ไปในที่สุด
- จากนั้นเคอร์เนลจะเริ่มสั่งให้ส่วนโปรแกรมที่ชื่อ init (/sbin/init) ทำงาน ซึ่งหลังจากที่ init สั่งให้โปรแกรมที่มีความจำเป็นให้เริ่มทำงานแล้ว ก็เป็นการเสร็จสิ้นกระบวนการบูต

7. เคอร์เนลจะ mount root filesystem แบบอ่านอย่างเดียว (read only) เพื่อตรวจสอบ file system หากพบข้อผิดพลาด ที่ทำให้ไม่สามารถ mount root filesystem ได้ ระบบจะไม่สามารถทำงานต่อได้และจะหยุดการทำงาน(แสดงข้อความเป็น kernel panic)ไปในที่สุด แต่ถ้าเมื่อการ mount สำเร็จ เคอร์เนลจะโหลดและสั่งรันโปรแกรมที่ชื่อ init (อยู่ที่ /sbin/init) ขึ้นมาทำงาน ซึ่งหลังจากที่ init ทำงานเรียบร้อยแล้ว ก็เป็นการเสร็จสิ้นกระบวนการบูตระบบ
8. เมื่อระบบไฟล์ mount ไม่สำเร็จ เคอร์เนลจะไม่สามารถทำงานได้ คอมพิวเตอร์จะหยุดการทำงาน
9. โพรเซส init จะอ่านข้อมูลคอนฟิกกูเรชันจาก /etc/inittab โดยข้อมูลในไฟล์ /etc/inittab จะเป็นตัวบอกว่าลินุกซ์จะเริ่มต้นการทำงานอย่างไร รวมทั้งเป็นตัวจัดการรันสคริปต์ต่างๆ เพื่อเตรียมความพร้อมในการทำงานด้วย รูปแบบของไฟล์คอนฟิกกูเรชัน (inittab) จะมีลักษณะดังนี้คือ

id:runlevel:action:process arguments

- id : เป็นเลเบลที่ไม่ซ้ำกันในไฟล์ เป็นตัวอักษรยาวได้ 2-4 ตัวอักษร
- runlevel : เป็นตัวระบุว่าคำสั่งในบรรทัดนี้จะมีผลกับ run level ใด ซึ่งสามารถใส่หลายๆ run level ได้
- action : ลักษณะของการรันโพรเซส
- process : โพรเซสที่ต้องการรัน

สำหรับ runlevel ขยายความในตารางที่ 4-1

ตารางที่ 4-1 runlevel

runlevel	ที่อยู่ของสคริปต์	คำอธิบาย
0	/etc/rc.d/rc0.d/	Shutdown หรือ Halt - ใช้หยุดการทำงานของระบบ
1	/etc/rc.d/rc1.d/	Single user - โหลดคอนฟิกน้อยที่สุด เพื่อให้ใช้งานได้คนเดียวเท่านั้น
2	/etc/rc.d/rc2.d/	ใช้งานได้หลายคนแต่ไม่สนับสนุนด้านเครือข่าย
3	/etc/rc.d/rc3.d/	ใช้งานได้หลายคนและผ่าน CLI เท่านั้น

4	/etc/rc.d/rc4.d/	สงวนไว้ใช้งาน(ไม่ได้ใช้งานในขณะนี้)
5	/etc/rc.d/rc5.d/	X Window - เป็นกราฟฟิกโหมด
6	/etc/rc.d/rc6.d/	Reboot - ใช้รีบูตระบบ
S หรือ s		ใช้งานได้เพียงคนเดียว สำหรับ Maintenance
M		Multiuser mode (Slackware)

สำหรับ action นั้น สามารถอธิบายได้ด้วยตารางที่ 4-2

ตารางที่ 4-2 action

Action	**คำอธิบาย**
respawn	รันโพรเซสใหม่ทุกครั้งเมื่อโพรเซสหยุดการทำงาน
wait	รันโพรเซสเพียงครั้งเดียว และให้ init รอจนกระทั่งโพรเซสสิ้นสุดการทำงาน
once	รันโพรเซสเพียงครั้งเดียว
boot	โพรเซสนี้รันในขณะบูต และให้ init ไม่สนใจค่าในส่วน id field
bootwait	โพรเซสนี้รันในขณะบูต และให้ init รอจน โพรเซสสิ้นสุดการทำงาน
off	จะไม่ทำการรันโพรเซสที่ระบุในบรรทัดนี้ เหมือนกับเป็นการ disable
ondemand	รันโพรเซสเมื่อมีการระบุ run level
initdefault	ตั้งค่า run level ที่ระบุเป็น default run level
sysinit	โพรเซสนี้จะถูกรันเพียงครั้งเดียวในขณะบูต
powerwait	โพรเซสจะถูกรันเมื่อได้รับสัญญาณ SIGPWR จาก UPS software โดย init จะรอจนกระทั่งโพรเซสสิ้นสุดการทำงาน
pawerfail	คล้ายกับ powerwait เพียงแต่ init จะไม่รอจนกระทั่งโพรเซสสิ้นสุดการทำงาน
powerokwait	โพรเซสจะถูกรันเมื่อได้รับสัญญาณ SIGPWR และมีคำว่า OK ในไฟล์ /etc/powerstatus โดยปกติเหตุการณ์นี้จะเกิดขึ้นเมื่อ UPS software แจ้งมาว่าเหตุการณ์ได้กลับสู่สภาวะปกติแล้ว
ctrlaltdel	โพรเซสนี้จะถูกเมื่อได้รับสัญญาณ SIGINT
kbrequest	โพรเซสนี้จะถูกเมื่อได้รับ KeyboardSignal จาก keyboard handler

10. ข้อมูลของไฟล์ /etc/inittab จะเป็นตัวบอกให้เคอร์เนลทำอะไรบ้าง ยกตัวอย่างเช่น

    id:3:initdefault:

จากบรรทัดด้านบน เป็นคำสั่งที่บอกให้ระบบทำงานใน run level ที่ 3 ซึ่งเป็น multiuser mode แต่สั่งงานผ่านทาง Text โหมด(CLI) เท่านั้น แต่ถ้ามีการติดตั้ง X Window ไว้ด้วย จะมี run level ที่เป็นดีฟอลต์คือ 5 อย่างไรก็ตามผู้ดูแลระบบสามารถแก้ไขค่านี้ให้เป็น run level ที่ต้องการได้ เช่น

แก้ไขคำสั่งให้ทำงานที่ run revel ที่ 5 แทน 3

id:5:initdefault:

11. จากนั้น init จะทำการเริ่มต้นระบบตามที่ระบุไว้ในสคริปต์ /etc/rc.d/rc.sysinit ที่ได้ระบุไว้ใน inittab

System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

โดย rc.sysinit มีฟังก์ชันการทำงานดังนี้

- ตั้งค่าตัวแปร $PATH
- ปรับแต่งเน็ตเวิร์ค
- โหลดค่า keyboard configuration, system font
- เริ่มการทำงานของส่วน swap
- เริ่มการทำงานส่วน USB controller
- ตรวจสอบ root filesystem ว่าต้องการการซ่อมแซมหรือไม่
- ทำการ remount root filesystem ใหม่ให้เป็นแบบ read-write
- ตรวจสอบ filesystem ว่าต้องการการซ่อมแซมหรือไม่
- เริ่มการทำงานของอุปกรณ์ plug and play
- เคลียร์ /etc/mtab (mounted filesystem table)
- เพิ่มค่า root filesystem ในไฟล์ /etc/mtab
- เริ่มต้นการทำ hard drive optimization
- เริ่มต้นการเซ็ตโควต้าของ user , group สำหรับ root filesystem
- ตั้งค่า hostname
- เตรียมพร้อมสำหรับการโหลดโมดูลต่างๆ
- ค้นหาโมดูลที่จำเป็นในการโหลด
- โหลดโมดูล sound
- เพิ่มอุปกรณ์ RAID และทำการ mount filesystem อื่นๆ
- เคลียร์ไฟล์ /etc/mtab, /etc/fastboot, /etc/nologin
- ลบล็อกไฟล์ของ UUCP
- ลบ stale subsystem files
- ลบ stale pid files
- เริ่มการทำงานของ serial port
- ตั้งค่าของ SCSI tape (ถ้ามี)

12. จากนั้นลินุกซ์จะทำการรันสคริปต์ภายใต้โฟลเดอร์ /etc/rc.d/rc ตามด้วย option คือ level ที่ระบุ ดัง configuration ใน /etc/inittab

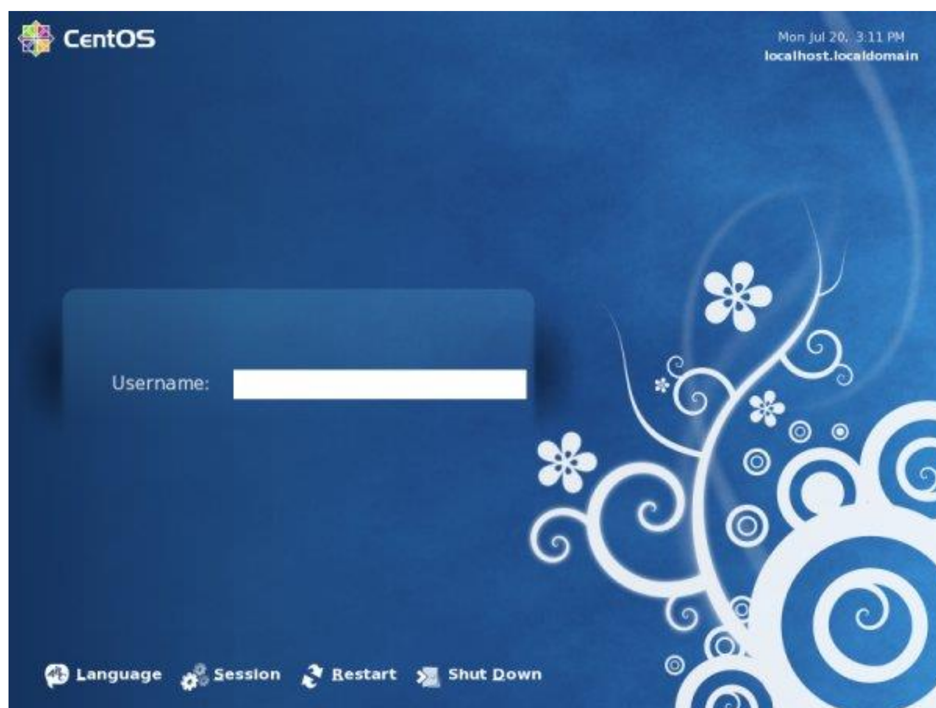
```
l0:0:wait:/etc/rc.d/rc0.d/
l1:1:wait:/etc/rc.d/rc1.d/
l2:2:wait:/etc/rc.d/rc2.d/
l3:3:wait:/etc/rc.d/rc3.d/
l4:4:wait:/etc/rc.d/rc4.d/
l5:5:wait:/etc/rc.d/rc5.d/
l6:6:wait:/etc/rc.d/rc6.d/
```

เช่นเมื่อถูกสั่งให้รันใน run level 3 สคริปต์ rc จะทำการตรวจสอบว่าไดเรคทอรี /etc/rc.d/rc3.d/ ว่ามีไฟล์อยู่หรือไม่ หากมีอยู่ก็จะทำงาน โดยจะรันสคริปต์ที่อยู่ภายใต้ /etc/rc.d/rc3.d/ และมีชื่อไฟล์ขึ้นต้นด้วยตัวอักษร S (uppercase) โดยจะส่งคำสั่งไปยังสคริปต์ดังกล่าวด้วยออปชัน start ผู้ใช้สามารถเปลี่ยนลำดับการทำงานก่อนหลังได้โดยเปลี่ยนค่าตัวเลขที่อยู่หลังตัวอักษร S หรือ K เช่น ไฟล์ S98httpd จะถูกสั่งให้ทำงานก่อนไฟล์ S99mysql ซึ่งหากต้องการให้ mysql ถูก start ก่อนก็สามารถเปลี่ยนชื่อไฟล์จาก S99mysql เป็นตัวเลขอื่นที่น้อยกว่า S98 และไม่ซ้ำกับไฟล์อื่นที่มีอยู่ในไดเรกทอรีเดียวกัน

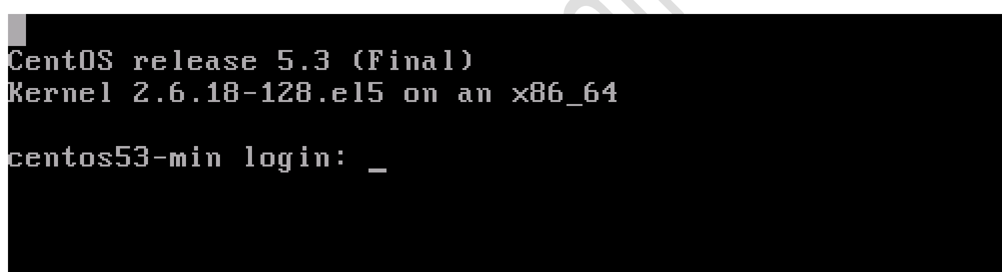
13. จากนั้น init จะทำการรันไฟล์ /etc/rc.local ซึ่งถูกรันเมื่อระบบถูกบูตหรือรันใน run level 2, 3 หรือ 5 โดยปกติมักจะใช้เพื่อใส่คำสั่งที่ต้องการให้ทำงานในการบูตแต่ละครั้ง
14. สุดท้ายก็จะรันสคริปต์ในไฟล์ /etc/rc.serial โดยปกติแล้วจะรันเมื่อสิ้นสุดกระบวนการใน run level 1 หรือ 3 เพื่อทำหน้าที่เริ่มต้นการทำงานของ serial port
15. สั่งรันเทอร์มินอลและ shell script เพื่อให้ผู้ใช้งานสั่งงานระบบต่อไป

การ Login

เมื่อลินุกซ์พร้อมทำงาน จะปรากฏหน้าต่างให้ทำการ Login เข้าระบบ โดยปกติจะมี 2 แบบ คือ แบบกราฟฟิกโหมด (Graphic Mode) และแบบ Text โหมด ดังรูป



รูปที่ 4-4 การ Login แบบกราฟฟิก



รูปที่ 4-5 การ Login แบบ Text โหมด

การชัตดาวน์ระบบ

กระบวนการชัตดาวน์มีความสำคัญต่อระบบลินุกซ์เป็นอย่างยิ่ง เพราะหากข้ามขั้นตอนหรือใช้คำสั่งผิดก็อาจจะทำให้เกิดความเสียหายต่อ file system ได้ ทั้งนี้เพราะลินุกซ์ทำงานโดยดึงข้อมูลจาก cache ซึ่งจะเก็บข้อมูลลงดิสก์เป็นช่วงเวลา หากเครื่องคอมพิวเตอร์ถูกปิดเครื่องไปโดยตรงก็อาจจะทำให้ข้อมูลใน cache ไม่ถูกบันทึกลงในไฟล์ซึ่งอาจจะทำให้ไฟล์มีปัญหาได้

นอกจากนี้ลินุกซ์ยังเป็นระบบปฏิบัติการที่เป็น multitask ซึ่งหากทำการ shutdown อย่างผิดขั้นตอน เช่น การกดปุ่มสวิตช์ power off โดยตรง เป็นต้น อาจจะส่งผลให้โพรเซสหลายๆ ตัวซึ่งทำงานใน background mode สูญเสียข้อมูล ซึ่งอาจจะเกิดความเสียหายต่อระบบได้ ก่อนทำการ shutdown ผู้ดูแลระบบจะต้องปิดการทำงานของโปรแกรมต่างๆ ที่รันอยู่ และใช้คำสั่ง shutdown ซึ่งจะส่งข้อความแจ้งเตือนไปยังผู้ใช้ที่ล็อกอินอยู่ในระบบให้ทราบว่า เครื่องกำลังจะ shutdown และ

แจ้งให้ผู้ใช้ทำการปิดการทำงานของโปรแกรมที่ใช้อยู่ พร้อมทั้งล็อกเอาต์ออกจากระบบ คำสั่งที่ใช้ในการชัตดาวน์ระบบคือ /sbin/shutdown ซึ่งสามารถใช้ได้หลายรูปแบบ เช่น

ให้ชัตดาวน์ระบบทันที

shutdown -h now

ให้ชัตดาวน์ระบบในอีก 10 นาที

shutdown -h +10

ให้ชัตดาวน์ระบบในอีก 10 นาที พร้อมกับส่งข้อความแจ้งผู้ใช้ทุกคนเป็นระยะๆ

shutdown -h +10 'System is going down for maintenance in 10 minutes'

เมื่อระบบถึงเวลาที่ต้องชัตดาวน์จริงๆ file system ทุกแห่งจะถูก unmount (ยกเว้น root) พร้อมกับ kill user process ทิ้งไป, ชัตดาวน์ daemon, file system ทุกที่จะถูก unmount และทุกสิ่งก็จะหยุดการทำงานโดยสิ้นเชิง มีคำสั่งที่สามารถใช้ชัตดาวน์ระบบ เช่นเดียวกับคำสั่ง shutdown คือคำสั่ง init 0 และ halt ซึ่งทั้งสองคำสั่งให้ผลเช่นเดียวกันกับ shutdown แต่ควรใช้คำสั่ง shutdown จะดีกว่าเพราะมีส่วนของการเตือนผู้ใช้ที่ยังล็อกออนอยู่ในระบบก่อนที่จะชัตดาวน์จริง

การรีบูตระบบ

การรีบูตระบบสามารถทำได้โดยใช้คำสั่ง shutdown -r now หรือ init 6 นอกเหนือจากการพิมพ์คำสั่งแล้ว หากไม่มีการแก้ไขค่าดีฟอลต์ใน /etc/inittab ผู้ใช้ยังสามารถกดปุ่ม ctrl-alt-del พร้อมกันเพื่อรีบูตระบบได้อีกด้วย

สรุปท้ายบท

การบวนการเริ่มต้นทำงานของคอมพิวเตอร์และระบบปฏิบัติการลินุกซ์เริ่มต้นจากเปิดเครื่องคอมพิวเตอร์ → BIOS ตรวจสอบอุปกรณ์ฮาร์ดแวร์ → BIOS ค้นหาอุปกรณ์ที่มีหน้าที่ในการบูตระบบปฏิบัติการ → ใน MBR ของฮาร์ดดิสก์เก็บโปรแกรม boot loader ซึ่งมีข้อมูลรายของพาร์ทิชันที่เก็บระบบปฏิบัติการ → โหลด LILO/GRUB เพื่อเลือกแล้วว่าจะบูตจากพาร์ทิชันใด → โหลดเคอร์เนลตรวจสอบฮาร์ดแวร์ mount root filesystem ตรวจสอบความถูกต้องของไฟล์ อื่นๆ → เชลล์ รอรับการใช้งานจากผู้ใช้

คำถามท้ายบท

1. จงอธิบายการทำงานของ BIOS มาพอเข้าใจ
2. จงอธิบายความสำคัญของ MBR และหน้าที่มาพอเข้าใจ
3. จงอธิบายว่า LILO และ GRUB คืออะไร และมีหน้าที่อย่างไร
4. ขบวนการ mount file system คืออะไร
5. kernel panic คืออะไร และมีสาเหตุมาจากอะไร
6. สคริปต์ /sbin/init ทำหน้าที่อะไร
7. สคริปต์ /etc/inittab ทำหน้าที่อะไร
8. สคริปต์ /etc/rc.d/rc.sysint ทำหน้าที่อะไร
9. จงอธิบายไดเรคทอรี /etc/rc.d/rc1.d, /etc/rc.d/rc2.d, /etc/rc.d/rc3.d, /etc/rc.d/rc4.d, /etc/rc.d/rc5.d ทำหน้าที่อะไร
10. จงอธิบายไฟล์ /etc/rc.local ว่ามีหน้าที่ทำอะไร
11. จงอธิบายการทำงานของ getty ว่ามีหน้าที่อย่างไร
12. ให้อธิบายว่าวิธีการ shutdown ลินุกซ์ สามารถทำได้กี่วิธี อะไรบ้าง พร้อมอธิบาย
13. ให้อธิบายว่าวิธีการ restart ลินุกซ์ สามารถทำได้กี่วิธี อะไรบ้าง พร้อมอธิบาย



บทที่ 5

คำสั่งยูนิกซ์/ลินุกซ์

วัตถุประสงค์

1. สามารถอธิบายคำสั่งบนระบบปฏิบัติการยูนิกซ์/ลินุกซ์ได้
2. สามารถประยุกต์ใช้คำสั่งในสถานการณ์ต่างๆ เพื่อใช้แก้ปัญหาได้
3. สามารถอธิบายการเขียนโปรแกรมเชลล์สคริปต์ได้

บทที่ 5
คำสั่งยูนิกซ์/ลินุกซ์

บทนำ

ในบทนี้อธิบายถึงการใช้คำสั่งบนระบบปฏิบัติการยูนิกซ์และลินุกซ์ เนื่องจากลีนุกซ์ถูกพัฒนาขึ้นมาโดยอาศัยรูปแบบส่วนใหญ่มาจากยูนิกซ์ ดังนั้นหากเรียนรู้ยูนิกซ์เราก็จะรู้ลีนุกซ์ด้วยหรือเรียนรู้ลินุกซ์ก็จะรู้ยูนิกซ์ด้วยเช่นเดียวกัน

เริ่มต้นการใช้งาน

เมื่อเราทำการล็อกอินเข้ามาใช้งานในระบบ เราสามารถจะติดต่อกับเครื่องคอมพิวเตอร์โดยผ่านโปรแกรมที่เรียกว่า "เชลล์" โดยที่เชลล์จะทำหน้าที่ตีความคำสั่งที่ได้รับจากผู้ใช้ส่งต่อไปยังคอมพิวเตอร์ และรับผลลัพธ์จากคอมพิวเตอร์มาแสดงให้กับผู้ใช้อีกทีหนึ่ง นั่นคือเชลล์จะทำหน้าที่เป็นตัวกลางคอยประสานงานการใช้งานระหว่างผู้ใช้และคอมพิวเตอร์ ในดอส(DOS: Disk Operating System) จะมีโปรแกรมอยู่ตัวหนึ่งที่ทำหน้าที่คล้ายกับเชลล์ ก็คือโปรแกรม COMMAND.COM แต่มีความซับซ้อนน้อยกว่าเชลล์ในยูนิกซ์มาก เราสามารถจะเขียนโปรแกรมให้ใช้งานกับเชลล์ได้ ซึ่งจะเรียกว่า เชลล์สคริปต์ (shell script) จะคล้ายกับ BATCH FILE บนดอสแต่มีประสิทธิภาพสูงกว่า [25] เชลล์บนยูนิกซ์จะมีที่นิยมใช้กันอยู่สองแบบคือ Bourne shells ซึ่งถูกเรียกตามผู้คิด คนแรกก็คือ Steven Bourne และเชลล์อีกแบบก็คือ C shells โดยผู้เขียนคนแรก ก็คือ Bill Joy (เป็นคนเขียน vi บนยูนิกซ์ด้วยเช่นกัน) ปัจจุบัน Bill Joy ทำงาน ให้กับบริษัท Sun Microsystems สำหรับเชลล์ที่นิยมใช้งานกันอยู่บนลีนุกซ์ก็คือ bash (Bourne Again Shell) ซึ่งเป็นเชลล์ที่พัฒนามาจาก Bourne shells [26] โดย Free Software Foundation มีการเพิ่มความสามารถบางอย่างจาก Bourne shells และใส่คุณสมบัติบางอย่างของ C shells เพิ่มเติมเข้ามาด้วย สำหรับในหนังสือเล่มนี้จะอ้างอิงกับ bash เป็นหลัก เมื่อทำการล็อกอินเข้าสู่ระบบปฏิบัติการได้สำเร็จ bash จะแสดงพร้อมพต์ (ใช้สัญญลักษณ์ $ หรือ #) ออกมารอรับคำสั่ง นั่นคือ ลีนุกซ์พร้อมรอรับคำสั่งแล้วนั่นเอง ดังตัวอย่าง

[root@localhost ~]#

หรือ

[root@localhost ~]$

เมื่อต้องการตรวจสอบว่าใช้เชลล์ตัวใดทำงานอยู่คือ ให้ใช้คำสั่ง echo $SHELL

```
#echo $SHELL
/bin/bash
#
```

5.1 ความรู้พื้นฐานเกี่ยวกับยูนิกซ์และลินุกซ์ที่ควรทราบ

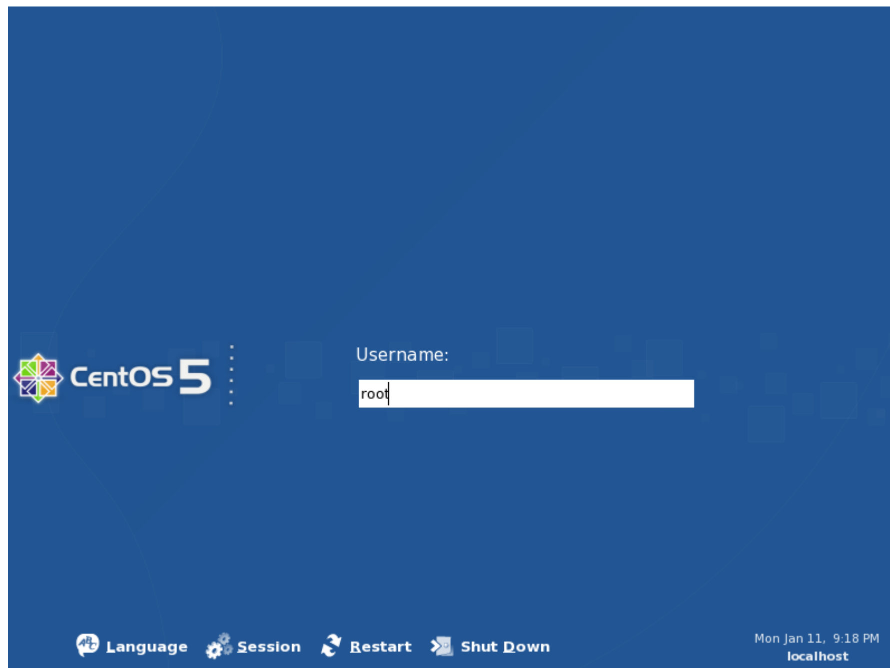
1. ยูนิกซ์เป็นระบบปฏิบัติการแบบ Multi User และ Multi Tasking ซึ่งแตกต่างจาก Windows ที่เป็นระบบปฏิบัติการแบบ Multi Tasking แต่ไม่เป็น Multi User กล่าวคือ ณ เวลาหนึ่งๆ บนระบบ Unix จะมีผู้ใช้งานเครื่องคอมพิวเตอร์ได้มากกว่า 1 คนพร้อมกัน ทำให้ Unix มีระบบการจัดการ Permission และระบบรักษาความปลอดภัยของข้อมูลดีกว่าและซับซ้อนกว่า DOS/Windows
2. ระบบ File System ของ Unix นั้นจะเป็นระบบ Single Root กล่าวคือจะมี Logical Driver เพียง Drive เดียวเท่านั้น และกรณีมี Harddisk หลายตัวหรือหลาย Partition แต่ละ Partition จะถูกกำหนดให้เป็นเพียง Directory ย่อยของระบบ ซึ่งจะต่างกับ DOS/Window ที่เป็นระบบ Multiple Root ที่จะแยก Drive / Parition ตามตัวอักษร เช่น A: , C: เป็นต้น
3. เนื่องจาก Unix เป็นระบบปฏิบัติการที่พัฒนาด้วยภาษา C ดังนั้นชื่อต่างๆ บน Unix จึงมีลักษณะเป็น Case-sensitive เช่น กรณีไฟล์ข้อมูลชื่อ MyFile กับ myfile จะเป็นไฟล์ข้อมูลคนละชื่อกัน
4. ระบบ Permission ของ Unix จะแบ่งเป็น 3 ระดับคือ ระดับเจ้าของ (User หรือ Owner) ระดับกลุ่ม (Group) และ ระดับบุคคลอื่น (Other) โดยในแต่ละระดับจะแบ่งออกเป็นสิทธิในการประมวลผล (execute) การอ่าน (read) และ การเขียน (write) ทั้งรายละเอียดเพิ่มเติมให้ดูจากคำสั่ง chmod
5. กรณีที่ผู้ใช้กระทำคำสั่งใดคำสั่งหนึ่งผิดพลาด Unix สามารถที่จะขัดจังหวะ (Interrupt) เพื่อยกเลิกการทำงานของคำสั่งหรือโปรแกรมนั้นๆ ได้โดยการกด CTRL + C
6. มาตรฐานของระบบ Keyboard บนเครื่อง Unix บางเครื่องอาจจะแตกต่างกับมาตรฐาน Keyboard บนเครื่องที่เราใช้อยู่ ดังนั้นในบางกรณี เช่น การใช้งานเครือข่ายระยะไกล (remote) จากเครื่องอื่นเข้าสู่ระบบ Unix อาจจะไม่สามารถใช้ Key บางอย่าง ตามปกติได้ เช่น backspace ดังนั้นเพื่ออำนวยความสะดวกให้เราสามารถใช้ backspace ได้ตามปกติจึงต้องมีการ map key ใหม่ด้วยการเรียกคำสั่ง stty erase [backspace] เป็นต้น

5.2 การ Login

การ Login เข้าระบบปฏิบัติการลินุกซ์สามารถทำได้ 3 แบบหลักๆ คือ

1. การ Login ด้วยกราฟฟิก

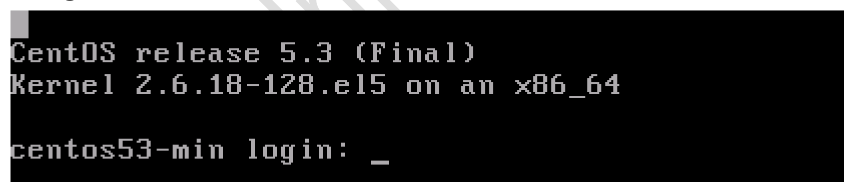
การ Login ด้วยวิธีการนี้ ผู้ใช้งานจะสามารถสั่งงานลินุกซ์ผ่านกราฟฟิก ถ้าต้องการออกจากระบบให้เลือกที่เมนู Logout



รูปที่ 5-1 Login ด้วยกราฟฟิกโหมด

2. การ Login ด้วย Text โหมด

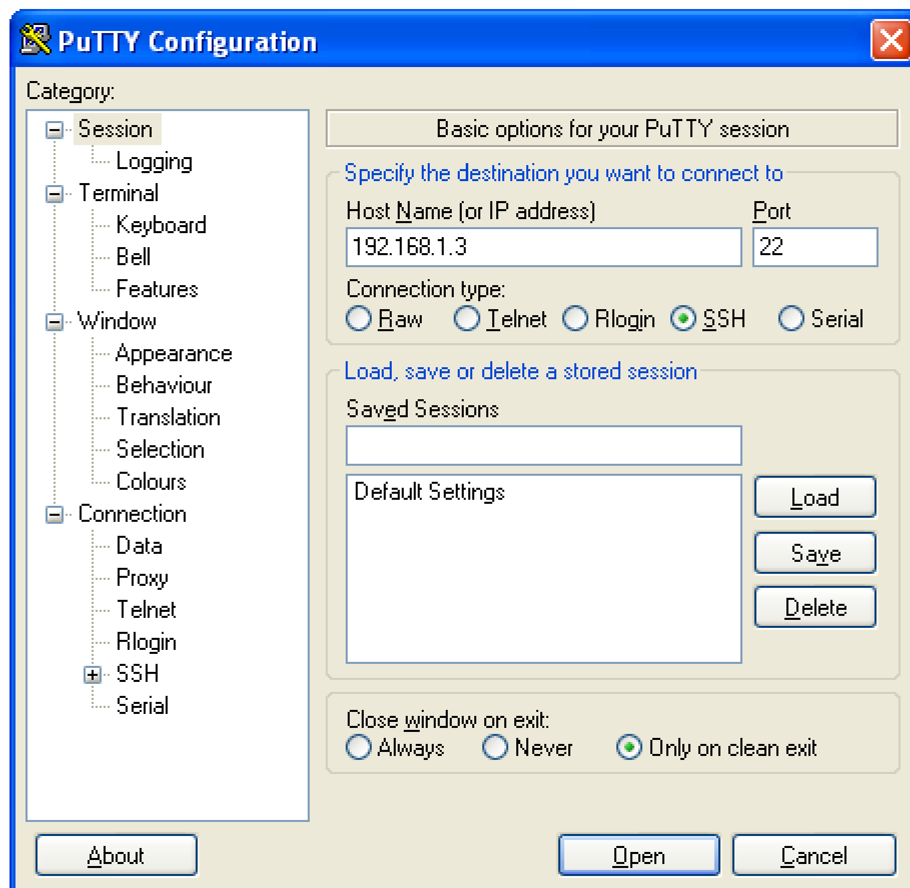
การสั่งงานลินุกซ์ทั้งหมดจะผ่านทาง command line เมื่อต้องการออกจากระบบให้ใช้คำสั่ง **exit** หรือ **logout**



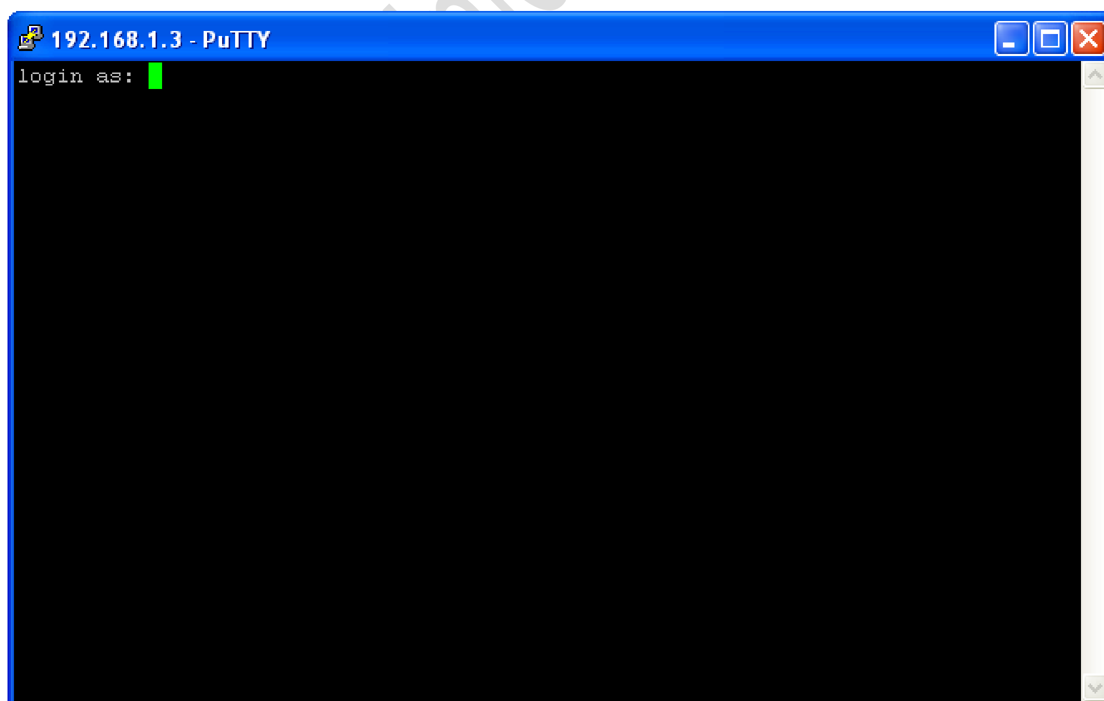
รูปที่ 5-2 Login ด้วย Text โหมด

3. การ Login ผ่าน Remote Login (Secure Shell: SSH)

ผู้ใช้งานสามารถ Login และใช้งานลินุกซ์จากที่ใดก็ได้โดยการ remote เข้ามาสั่งงานลินุกซ์ผ่านทางระบบเครือข่าย ซึ่งช่วยให้ผู้ดูแลระบบสามารถควบคุมและสั่งการลินุกซ์ได้สะดวก โดยไม่จำเป็นต้องควบคุมอยู่หน้าเครื่อง



รูปที่ 5-3 ระบุไอพีแอดเดรสที่ต้องการ Remote Login ผ่าน Secure Shell(Putty)



รูปที่ 5-4 Login ด้วย Secure Shell(Putty)

5.3 คำสั่งบนระบบปฏิบัติการลินุกซ์ [27-28, 32]

สำหรับการสั่งงานลินุกซ์ผ่าน command line(CLI) จะมีรูปแบบการใช้คำสั่งดังนี้

command [-option] [parameter]

command คือคำสั่ง

-option เป็นการระบุถึง Option เพื่อเลือกการทำงานแบบใดแบบหนึ่ง

parameter เป็นข้อมูลหรือตัวแปรเพิ่มเติม เพื่อให้คำสั่งมีความสมบูรณ์มากขึ้น

ข้อความใน [ ] เป็นส่วนที่มีหรือไม่มีก็ได้

คำสั่งบนลินุกซ์เป็นแบบ case sensitive คืออักษรตัวเล็กตัวใหญ่จะมีความแตกต่างกัน เช่น a กับ A จะมีความหมายต่างกัน

ตัวอย่างเช่น **#ls -l /usr/local**

ls คือ command

-l คือ option เพื่อให้แสดงผลรายละเอียดของไฟล์และไดเรคทรอรีแบบละเอียด

/usr/local คือ ตัวแปร เพื่อบอกให้คำสั่งดังกล่างแสดงรายละเอียดภายในไดเรคทรอรีชื่อว่า /usr/local เท่านั้น

Path คือที่อยู่ของ File หรือ Directory ในระบบยูนิกซ์/ลินุกซ์ แบ่งเป็น 2 ชนิดคือ

Absolute Path (เป็นการอ้างอิงแบบสมบูรณ์หรือแบบยาว) โดยเริ่มอ้างจาก / (เรียกว่า root directory เป็น Directory เริ่มต้นของระบบไฟล์ในยูนิกซ์) ตัวอย่างเช่น เมื่อต้องการอ้างอิงแบบ absolute ของไฟล์ที่เก็บรหัสผู้ใช้ไว้สามารถอ้างอิงได้โดย /etc/passwd

Relative Path (เป็นการอ้างอิงแบบอนุโลมหรือแบบสั้นกระชับ) อ้างอิงจาก Directory ที่ทำงานอยู่ในปัจจุบัน (Working Directory) การอ้างอิงแบบ Relative มีสัญลักษณ์แทน Directory ปัจจุบนเป็น . (จุด) และ Directory ทีอยู่เหนือขึ้นไปหนึ่งระดับ (Parent Directory) แทนด้วย .. (จุดสองจุดติดกัน) เช่น ./root, cd ..

wildcard ใช้สำหรับแทนชื่อไฟล์หรือไดเรคทรอรี ใช้ * แทนอักษรอะไรก็ได้ กี่ตัวก็ได้ และ ? แทนตัวอักษรเพียง 1 ตัว อะไรก็ได้

working directory เป็นการอ้างอิงถึงไดเรคทรอรีที่กำลังทำงานอยู่ขณะนั้นๆ ถ้าการอ้างอิงใดๆ ไม่เป็นแบบ absolute path จะถือว่าเป็นแบบการอ้างอิงใน working directory อัตโนมัติ(สามารถแสดง working directory ด้วยคำสั่ง pwd)

su คำสั่งบนลินุกซ์บางคำสั่งที่เป็นคำสั่งที่เกี่ยวข้องกับระบบจำเป็นต้องใช้สิทธิ์ของ root ดังนั้นจะต้องออกคำสั่งเพื่อเปลี่ยนสิทธิ์เป็น root ด้วยคำสั่ง su แล้วตามด้วยคำสั่งที่ต้องการใช้ เช่น

su /etc/init.d/sshd start

5.3.1 กลุ่มคำสั่งเกี่ยวกับ File/Directory Basics

คำสั่ง ls

File/Directory Basics : ls	
คำสั่ง	ls เป็นคำสั่งที่ใช้สำหรับแสดงรายชื่อไฟล์ ไดเร็คทอรี่ ในรูปแบบต่างๆ (คล้ายการทำงานของ dir ใน dos) ย่อมาจากคำว่า list
Syntax	ls [*options*] [*names*]
Example	# ls # ls -l # ls -al /etc/init.d

options :

-1(เลขหนึ่ง), --format=single-column แสดงรายชื่อไฟล์ทั้งหมดเพียงคอลัมภ์เดียว

-a, --all แสดงรายชื่อไฟล์และไดเรคทรอรีทั้งหมด รวมถึงไฟล์ที่ซ่อน และไฟล์แบบ ., .. ด้วย

-b, --escape แสดงรายชื่อไฟล์และไดเรคทรอรีแบบเรียงลำดับตัวอักษร

-c, --time-ctime, --time=status แสดงรายชื่อไฟล์และไดเรคทรอรี โดยเรียงจากระยะเวลาการการแก้ไข (แก้ไขล่าสุดแสดงก่อน)

--color =*when* แสดงรายชื่อไฟล์และไดเรคทรอรี โดยกำหนดรูปบบของสีให้ตามค่า when โดย when มีค่าเป็น auto, naver, always

-d, --directory แสดงรายชื่อไดเรคทรอรี 1 ไดเรคทรอรีเท่านั้น

-f แสดงรายชื่อไฟล์และไดเรคทรอรีปัจจุบัน โดยไม่มีการจัดเรียงข้อมูล

--full-time แสดงรายชื่อไฟล์และไดเรคทรอรี โดยแสดงเวลาแบบเต็ม

-g แสดงรายชื่อไฟล์และไดเรคทรอรีแบบยาวและไม่แสดงชื่อเจ้าของไฟล์

-h แสดงรายชื่อไฟล์และไดเรคทรอรี พร้อมขนาดของไฟล์ที่ใช้ มีหน่วยเป็นกิโลไบต์ โดยต้องใช้คู่กับออพชัน –l

--help แสดงข้อความช่วยเหลือการใช้คำสั่ง ls

-i, --inode แสดงรายชื่อไฟล์และไดเรคทรอรี พร้อมหมายเลข index number ของแต่ละไฟล์

-k, --kilobytes inode แสดงรายชื่อไฟล์และไดเรคทรอรี มีขนาดเป็นกิโลไบต์

-l, --format=long, --format=verbose แสดงผลลัพธ์แบบละเอียด (Long Format) ซึ่งจะแสดง Permission ของไฟล์ด้วย

-m, --format=commas แสดงผลลัพธ์แบบคอลัมภ์ โดยคั่นด้วย ,

-n, --numeric-uid-gid แสดงผลคล้าย –l แต่เปลี่ยนชื่อและกลุ่มของผู้ใช้จากตัวอักษร เป็นตัวเลขแทน เช่น ผู้ใช้ชื่อ root จะมีตัวเลขเป็น 0 และกลุ่ม root จะมีตัวเลขเป็น 0 เช่นเดียวกัน

-o แสดงผลเหมือน –l แต่ไม่แสดงในส่วนของ group

-p, --filetype, --indicator-style=file-type แสดงผลของไดเรคทรอรีด้วยเครื่องหมาย /

-r, --reverse แสดงผลโดยเรียงรายชื่อย้อนลำดับ (จาก z ไปยัง a)

-s, --size แสดงขนาดของไฟล์เป็นจำนวน block

-t, --sort=time แสดงข้อมูลโดยเรียงลำดับจากไฟล์ที่ถูกแก้ไขล่าสุด หรือใหม่ที่สุดขึ้นก่อน

--version แสดงเวอร์ชันของ ls

-w, --width=*n* แสดงผลแบบกำหนดคอลัมภ์ โดย n คือค่าคอลัมภ์ที่ต้องการแสดงผล

-A, --almost-all แสดงผลคล้าย –a แต่ไม่แสดงไฟล์ประเภท ., ..

-F แสดงสัญลักษณ์ / หลังไดเรคทรอรี และแสดง * หลังไฟล์ข้อมูลที่ execute ได้

-R, --recursive แสดงไดเรคทรอรีแบบ recursive

-S, --sort=size แสดงโดยเรียงลำดับตามขนาดของไฟล์ (จัดเรียงตามขนาดไฟล์ใหญ่ไปยังขนาดเล็ก)

จากรูปที่ 5-5 เป็นการใช้คำสั่ง ls –al แสดงรายการไฟล์และไดเรคทรอรีแบบละเอียด โดยแสดงทั้งไฟล์ที่ซ่อนทั้งหมดด้วย (ไฟล์ชนิด . และ ..) ซึ่งสามารถอธิบายความหมายของแต่ละคอลัมภ์ได้ดังนี้

```
[root@localhost ~]# ls -al
total 440
dr          23                  4096             9:56   .
dr          24                  4096             8:29   ..
-rw-r--r--   1  root            81     Nov 1  15:29   a
drwx------   3  abc    test     4096   Nov 15 18:27   abc
-rw-r--r--   1  root   root    32256   Nov 15 18:38   abc.tar.gz
-rw-------   1  root   root     1155   Nov  8 16:09   anaconda-ks.cfg
-rw-r--r--   1  root   root        0   Jan 13 19:39   a.txt
-rw-r--r--   1  root   root       14   Nov 15 18:51   b
-rw-------   1  root   root    13992   Jan 13 14:08   .bash_history
-rw-r--r--   1  root   root       24   Jan  6  2007   .bash_logout
-rw-r--r--   1  root   root      191   Jan  6  2007   .bash_profile
-rw-r--r--   1  root   root      176   Jan  6  2007   .bashrc
-rw-r  r--   1  r      root     1 81   Nov 22 12:08   bk
```


รูปที่ 5-5 แสดงการใช้คำสั่ง ls –al

- ชนิดและสิทธิ์ จะบ่งบอกประเภทของไฟล์นั้นๆ ว่าเป็นไฟล์ชนิดใด โดยจะกำหนดเป็นอักษร 10 ตัว โดยอักขระตัวแรกจะบอกถึงชนิดของไฟล์หรือไดเรคทอรี เช่น d

หมายถึงไดเรคทรอรี, l หมายถึงลิงค์ไฟล์, - หมายถึง text ไฟลหรือไฟล์ข้อมูลธรรมดา เป็นต้น จากตัวอย่างไฟล์ชื่อ abc เป็นไดเรคทอรี(มีสัญญลักษณ์ d เป็นตัวแรก) สำหรับอักษรอีก 9 ตัวถัดมา เป็นการกำหนดสิทธิ์การอ่าน การเขียน และการประมวลผล ซึ่งกำหนดได้โดยใช้คำสั่ง chmod

- จำนวนลิงค์ จะบอกให้ทราบถึงจำนวนไฟล์ที่ลิงค์เข้ากับไดเรคทอรีหรือไฟล์นี้ ถ้าเป็นไดเรคทอรีว่างจะมีค่าระบุไว้เป็น 2
- เจ้าของ จะบอกให้ทราบว่าใครเป็นเจ้าของไฟล์หรือไดเรคทอรีนี้ เช่น root
- ชื่อกลุ่ม บอกให้ทราบผู้ใช้อยู่ในกลุ่มใด
- ขนาด บอกให้ทราบถึงขนาดของไฟล์ว่ามีความจุเท่าไหร่ โดยแสดงเป็นไบต์ (Byte) แต่ถ้าเป็นไดเรคทอรีจะมีขนาดเท่ากับ 4096 ไบต์เสมอ
- วัน-เวลา จะแสดงวันและเวลาในการใช้งานไฟล์หรือไดเรคทอรีล่าสุด
- ชื่อไฟล์ หรือไดเรคทอรี แสดงชื่อไฟล์หรือไดเรคทรอรี

ตัวอย่างการใช้งาน :

```
# ls ~
a   abc.tar.gz     a.txt  bk  date.txt  error   install.log      list   result    t
```
แสดงรายการไฟล์และไดเรคทรอรี ภายใน home ไดเรคทรอรีของตนเอง

```
# ls /
bin boot dev etc home lib lost+found media misc mnt net opt proc root
sbin selinux srv sys tm tmp usr var
```
แสดงรายการไดเรคทรอรีและไฟล์ที่อยู่ภายในไดเรคทรอรีรูท (/)

```
# ls /usr/bin
```
แสดงรายการไดเรคทรอรีและไฟล์ ภายในไดเรคทรอรี /usr/bin

```
# ls -al
total 464
drwxr-x--- 24 root    root     4096 Jan 14 18:19 .
drwxr-xr-x 24 root    root     4096 Jan 15 04:35 ..
-rw-r--r--  1 root    root       81 Nov 15 15:29 a
drwx------  3 abc     test     4096 Jan 13 21:07 abc
-rw-r--r--  1 root    root    32256 Nov 15 18:38 abc.tar.gz
```
แสดงข้อมูลในไดเรคทรอรีปัจจุบันอย่างละเอียดพร้อมไฟล์ที่ซ่อนไว้ ภานในไดเรคทรอรีที่ทำงานในปัจจุบัน

```
# ls -alt
[root@localhost ~]# ls -alt
total 464
-rw-r--r--  1    root    root     3654 Jan 15 04:37 ls.txt
drwxr-xr-x 24 root    root     4096 Jan 15 04:35 ..
-rw-------  1    root    root    14391 Jan 14 18:19 .bash_history
```

drwxr-x--- 24 root root 4096 Jan 14 18:19 .
drwx------ 2 root root 4096 Jan 14 18:19 .gconfd
drwx------ 4 root root 4096 Jan 14 18:06 .gconf

แสดงรายชื่อไฟล์ และจัดเรียงไฟล์ตามเวลาที่ไฟล์ถูกใช้งานล่าสุดขึ้นก่อน (a=all, l=long listing, t=sort by modification time)

ls -al --sort=time

แสดงรายชื่อไฟล์เรียงตามเวลาแบบละเอียด โดยเรียงลำดับไฟล์ที่ถูกใช้งานล่าสุดขึ้นก่อน

ls –R

แสดงรายชื่อไฟล์ข้อมูลและเส้นทางที่จัดเก็บในไดเรคทรอรีปัจจุบัน แบบ Recursive

ls –ash

แสดงรายชื่อไฟล์และไดเรคทรอรี รวมถึงขนาดของไฟล์แบบ blocks

ls -lh

แสดงรายชื่อไฟล์และไดเรคทรอรี รวมถึงขนาดของไฟล์ในรูปแบบที่อ่านง่าย(kilobyte)

ls -F

a abc.tar.gz a.txt* bk date.txt error install.log list result t
t1.txt test/ t.txt abc/ anaconda-ks.cfg b c Desktop/ install/
install.log.syslog ls.txt sort.txt t1/ tar/ test1*

แสดงรายการไดเรคทรอรีและไฟล์ โดยใช้เครื่องหมาย / อยู่ท้ายแสดงว่าเป็นไดเรคทรอรี และใช้เครื่องหมาย * อยู่ท้ายแสดงว่าสามารถประมวลผลได้

ls -F --color=never

แสดงรายชื่อไฟล์และไดเรคทรอรี โดยไม่ใช้สีในการแสดงผล

ls -F --color=auto
ls -F --color=aways



คำสั่ง ls –F จะให้ผลลัพธ์ ซึ่งแสดงเป็นสีต่างๆ ดังนี้

สีน้ำเงิน = directory

สีดำ = ไฟล์ธรรมดา

สีเขียว = ไฟล์ที่สามารถ execute ได้

ลองทดสอบ :

```
# ls ../
# ls */
# ls -d */
# ls *[0-9]*
# ls -alt | more
# ls /nofile 2> /tmp/error
```

ดูคำสั่งอื่นประกอบ : *chmod, df, diff, du, file, stat, tree, ls --help และ man ls*

คำสั่ง cp

File/Directory Basics : **cp**	
คำสั่ง	**cp** เป็นคำสั่งที่ใช้สำหรับสำเนาไฟล์ข้อมูล (เหมือนกับคำสั่ง copy ใน dos) มาจากคำว่า copy
Syntax	**cp [*options*] *file1 file2*** **cp [*options*] *files directory***
Example	**# cp test1.txt test2.conf** **# cp test1.txt /usr/local** **# cp -rf /etc/ /export/backup**

options :

-a, --archive ข้อมูลที่สำเนาจะเหมือนกับต้นฉบับทุกประการ เหมือนกับการใช้ option –dpR

-b, --backup สำเนาข้อมูลแบบมีการสำรองข้อมูลไว้ด้วย เมื่อชื่อไฟล์ซ้ำกัน จะทำการสำรองเป็นไฟล์ชื่อเดียวกัน แต่มีส่วนขยายเป็น ~ เพิ่มเติมเพื่อบอกให้ทราบว่าไฟล์ข้อมูลที่สำรองซ้ำกัน หรือมีอยู่ก่อนหน้าที่จะทำการสำรอง

-d, --no-dereference สำเนาข้อมูลโดยไม่ต้องเปลี่ยน hard-link ไฟล์ที่สำเนาใหม่จะมี hard-link เชื่อมโยงไปยังไฟล์เดิม

-f, --force ถ้ามีไฟล์เหมือนกันในไดเรคทรอรีปลายทางจะทำการลบทิ้งและเขียนข้อมูลใหม่ทับทันที

-i, --interactive ทำการแจ้งเตือนก่อนทำการสำเนา ในกรณีที่มีไฟล์เดิมอยู่แล้ว

-l, --link ไม่ทำการสำเนาจริง เพียงแต่ทำการสร้าง hard links ไปยังไฟล์ใหม่เท่านั้น

-p, --preserve สำเนาโดยการสงวนคุณสมบัติของไฟล์เดิมไว้เช่น owner, group, permissions, and timestamps.

-r, -R, --recursive สำเนาข้อมูลทั้งโครงสร้างของไดเรคทรอรี หรือทั้ง tree ของไดเรคทรอรีที่ต้องการทั้งหมด

-s, --symbolic-link สร้าง symbolic links แทนการสำเนาข้อมูลจริง

-u, --update ไม่สำเนาข้อมูล เมื่อปลายทางมีข้อมูลเหมือนกันหรือใหม่กว่า

-v, --verbose ขณะทำการสำเนาข้อมูลจะพิมพ์ข้อมูลไฟล์ที่จะสำเนาทุกรายการ

ตัวอย่างการใช้งาน :

cp file1.txt newdir

สั่งสำเนาข้อมูลชื่อว่า file1.txt จากไดเรคทรอรีที่ทำงานอยู่ ไปยังไดเรคทรอรีชื่อว่า newdir

cp /home/public_html/mylog.txt /home/public_html/backup/mylog.bak

สำเนาข้อมูลชื่อว่า mylog.txt ในไดเรคทรอรี /home/public_html/ ไปยังไดเรคทรอรี

/home/public_html/backup/ ชื่อว่า mylog.bak

cp *.txt newdir

สำเนาทุกไฟล์ที่มีส่วนขยายเป็น .txt ไปยังไดเรคทรอรีชื่อว่า newdir

cp -r /home/hope/files/* /home/hope/backup

สำเนาทุกไฟล์ที่อยู่ในไดเรคทรอรี /home/hope/files ไปไว้ใน /home/hope/backup

cp file1 file2

สำเนาไฟล์ชื่อว่า file1 พร้อมกับเปลี่ยนชื่อใหม่เป็น file2

cp -b file1.php file2.php

สำเนาไฟล์ชื่อว่า file1.php เป็นแบบสำรองข้อมูลด้วย โดยมีชื่อว่า file2.php~

cp -R scripts scripts1

สำเนาไดเรคทรอรีชื่อว่า scripts ถ้ามีไดเรคทรอย่อยก็ทำการสำรองทั้งหมด โดยตั้งชื่อไดเรคทรอรีใหม่ว่า scripts1

cp /root/* .

สำเนาไฟล์ทุกไฟล์ที่อยู่ในไดเรคทรอรี /root มาไว้ที่ไดเรคทรอรีที่กำลังทำงานอยู่ปัจจุบัน (ยกเว้นไดเรคทรอรีจะไม่สำเนามาด้วย)

cp -a dir1 dir2

สำเนาไดเรคทรอรี dir1 เป็น dir2

cp -b file1.php file2.php

สำเนาไฟล์ชื่อว่า file1.php เป็นแบบสำรองข้อมูลด้วย โดยมีชื่อว่า file2.php~

cp -Rf /root .

สำเนาไฟล์ทุกไฟล์และไดเรคทรอรีที่อยู่ในไดเรคทรอรี /root มาไว้ที่ไดเรคทรอรีที่กำลังทำงานอยู่ปัจจุบัน

ลองทดสอบ :

```
# cp -i file1 file2
# cp -f file1 file2
# cp -s file1 file2
```

ดูคำสั่งอื่นประกอบ : *mv*

คำสั่ง mv

File/Directory Basics : **mv**	
คำสั่ง	**mv** เป็นคำสั่งที่ใช้สำหรับการย้ายไฟล์ข้อมูลและไดเรคทรอรี รวมถึงการเปลี่ยนชื่อด้วย มาจากคำว่า move
Syntax	**mv [*option*] *sources target***
Example	**# mv file1 file2** **# mv myfile.txt /usr/backup**

options :

-b ทำการสำรองไฟล์ก่อนทำการย้ายไฟล์ข้อมูล

--backup[=*type*] ทำการสำรองก่อนการย้ายแบบมีเงื่อนไขในการสำรอง คือ type ถ้า type มีค่าเป็น

none, off ไม่ทำการสำรองข้อมูล

numbered, t สร้างหมายเลขลำดับการสำรองให้

existing, nil สร้างหมายเลขการสำรองใหม่ เมื่อหมายเลขเดิมมีอยู่แล้ว

simple, never สำรองข้อมูลแบบธรรมดา

-f, --force เมื่อมีไฟล์อยู่แล้วจะสั่งให้ทำการย้ายไปทับทันที (บังคับให้ทำให้สำเร็จ)

--help อธิบายการใช้งาน mv

-i, --interactive สอบถามผู้ใช้งานก่อนการย้าย

--reply=*prompt* มีการยืนยันก่อนว่าจะทำการย้ายหรือไม่ คล้าย –i

--target-directory=*dir* ย้ายไฟล์หรือไดเรคทรอรีไปยังปลายทางตามที่ระบุ

-v, --verbose แสดงผลลัพธ์การย้ายทุกครั้ง

source คือชื่อไฟล์ต้นฉบับที่ต้องการย้ายหรือลบ

target คือชื่อไฟล์ปลายทางที่จะย้ายไป อาจจะเป็นชื่อเดิม หรือชื่อใหม่ก็ได้

ตัวอย่างการใช้งาน :

mv filename.tar /backup
ย้ายไฟล์ชื่อว่า filename.tar ไปไว้ในไดเรคทรอรี /backup

mv *.tar /backup
ย้ายไฟล์ทุกไฟล์ที่มีส่วนขยายเป็น .tar ไปไว้ในไดเรคทรอรี /backup

mv *.* /somedir/backup
ย้ายไฟล์ทุกไฟล์ไปไว้ในไดเรคทรอรี /somedir/backup

mv somedir/* dest/backup
ย้ายทุกไฟล์ทุกโฟลเดอร์ ใน somedir ไปที่ ไดเรคทรอรี dest/backup

mv OldName.txt NewName.txt
เปลี่ยนชื่อไฟล์จาก OldName.txt เป็น NewName.txt

mv myfile.txt newdirectory/
ย้ายไฟล์ชื่อ myfile.txt ไปไว้ในไดเรคทรอรีชื่อ newdirectory

mv myfile.txt ../
ย้ายไฟล์ชื่อ myfile.txt ไปไว้ในไดเรคทรอรีด้านบนอีก 1 ระดับ

mv scripts tmp
เปลี่ยนชื่อไดเรคทรอรีชื่อ script เป็น tmp

mv file1.txt tmp/file2.txt newdir
ย้ายไฟล์ชื่อ file1.txt จากไดเรคทรอรีปัจจุบันกับ file2.txt จากไดเรคทรอรี tmp ไปไว้ในไดเรคทรอรีชื่อว่า newdir

mv /usr/local/test/net.conf .

ย้ายไฟล์ชื่อ net.conf อยู่ที่ /usr/local/test/ มาไว้ที่ไดเรคทรอรีที่กำลังทำงานอยู่ **# mv -i file file2** ถามก่อนทำการเปลี่ยนชื่อ

ลองทดสอบ :

```
# mv * /home/alice/new/
# mv dir/dir/file4 ~
# mv file dir1
```

ดูคำสั่งอื่นประกอบ : *copy, cp, cpio, ln, rm, setfacl*

คำสั่ง rm

File/Directory Basics : **rm**	
คำสั่ง	**rm** เป็นคำสั่งที่ใช้สำหรับลบไฟล์ข้อมูล (เหมือน del ใน dos) มาจากคำว่า remove
Syntax	**rm *[options] [files|directory]***
Example	**# rm myfile** **# rm -rf myfile**

options :

-f, --force ลบไฟล์หรือไดเรคทรอรีทันที โดยไม่มีการแจ้งเตือนการลบ

-i, --interactive ถามยืนยันก่อนทำการลบ

-r, -R, --recursive ทำการลบข้อมูลใน directory ย่อยทั้งหมด

-v, --verbose พิมพ์รายการที่ลบทั้งหมด

-- ใช้เมื่อต้องการลบไฟล์ที่ขึ้นต้นด้วยอักษร –

files ชื่อไฟล์ที่ต้องการลบ

directory ชื่อไดเรคทรอรีที่ต้องการลบ

ตัวอย่างการใช้งาน :

rm myfile.txt ลบไฟล์ชื่อว่า myfile.txt **# rm -r directory** ลบไดเรคทรอรีชื่อว่า directory **# rm -r /tmp** ลบไดเรคทรอรี tmp และหากมีไดเรคทรอรีหรือไฟล์อื่นๆ อยู่ใน tmp ก็ลบทั้งหมดทิ้งด้วย **# rm -rf install** ลบไดเรคทรอรี install และหากมี ไฟล์หรือไดเรคทรอรีอะไรอยู่ในนั้น ก็ลบให้หมด และไม่ต้องมีการยืนยันการลบ **# rm -f *** ลบไฟล์ทั้งหมดในไดเรคทรอรีปัจจุบันโดยไม่มีการยืนยัน

rm -i dog cat pig
ลบไฟล์ชื่อว่า dog cat pig โดยต้องมีการยืนยันการลบ ด้วยคำว่า y หรือ yes จะทำการลบแต่ถ้า กดปุ่ม enter หรือ n, no จะไม่มีการลบเกิดขึ้น

info rm
คำสั่งแสดงคู่มือการใช้คำสั่ง rm

man rm
คำสั่งแสดงคู่มือการใช้คำสั่ง rm อย่างละเอียด

rm -help
คำสั่งแสดงการใช้คำสั่ง rm แบบย่อ

ลองทดสอบ :

```
# rm -ir tmp
# mv file1 /dev/null
# rm -r ./*important-file*
# rm -- -not-important*
# rm 'happy rock hacking.mp3'
# rm -rf ./*
```

ดูคำสั่งอื่นประกอบ : *rmdir, shred*

คำสั่ง ln

File/Directory Basics : **ln**	
คำสั่ง	**ln** เป็นคำสั่งสร้างไฟล์เชื่อมโยง (คล้ายการสร้าง shortcut ในวินโดวส์)
Syntax	***ln [options] existingfile newname***
Example	**# ln -s sourcefile destfile** **# ln -s /usr/local/httpd/sbin/httpd.sh /etc/init.d/rc.d/rc3.d/S99httpd**

options :

-f, --force สร้างการเชื่อมโยงลิงค์ โดยเขียน permission ใหม่ทับทันที

-n, --no-dereference สร้างการเชื่อมโยง แต่ไม่แสดงสัญลักษณ์การเชื่อโยงให้เห็น

-s, --symbolic สร้างการเชื่อมโยงลิงค์ และสร้าง symbolic link ให้ด้วย

existingfile ไฟล์ต้นฉบับที่ต้องการสร้างลิงค์

newname ชื่อไฟล์ที่ต้องการสร้างลิงค์ใหม่

directory ชื่อไดเรคทอรีที่ต้องการสร้างลิงค์ใหม่

ตัวอย่างการใช้งาน :

ln /etc/init.d/nfs ./tnfs
สร้างลิงค์เชื่อมโยงจากไฟล์ชื่อ nfs ที่อยู่ในไดเรคทรอรี /etc/init.d/ ให้เป็นชื่อ tnfs โดยไม่มีการสร้างสัญลักษณ์เชื่อมโยงลิงค์ให้เห็น

ln -s file1 file2
สร้างลิงค์เชื่อมโยงจากไฟล์ชื่อ file1 ไปยัง file2 และแสดงสัญลักษณ์การเชื่อมโยง (ใช้

คำสั่ง ls –l จะปรากฏอักษร l อยู่ด้านหน้าสุดของไฟล์ file2)
rm file2
ลบลิงค์ที่สร้างชื่อไฟล์ file2 ทิ้ง

ลองทดสอบ :

```
# ln -s /home/user1 /var/www/public_html
# rm rf /var/www/public_html
```

ดูคำสั่งอื่นประกอบ : *chmod, link, ls*

คำสั่ง cd

File/Directory Basics : **cd**	
คำสั่ง	**cd** เป็นคำสั่งที่ใช้สำหรับเปลี่ยน หรือย้ายไดเรคทรอรี (cd มาจากคำว่า change directory)
Syntax	**cd *[directory]***
Example	**# cd /root** **# cd ~**

options :

-directory คือไดเรคทรอรีที่ต้องการย้ายเข้าไปทำงาน โดยอ้างได้ 2 แบบคืออ้างแบบสมบูรณ์หรือเต็ม(absolute path) เช่น cd /home/test และแบบที่ 2 คือแบบย่อ (relative path) เช่น cd ~ ซึ่งที่อยู่จริงคือ /home/test

cd .. เป็นคำสั่งที่ย้ายไดเรคทรอรีขึ้นไปจากไดเรคทรอรีที่กำลังทำงานอยู่ 1 ชั้น (cd จะต้องเว้นวรรคและตามด้วย ..)

cd ~ เป็นคำสั่งที่ย้ายไปยังไดเรคทรอรีของตนเอง เช่น ถ้า login ด้วย root ถ้าใช้คำสั่งดังกล่าวจะกลับไปที่ /root

cd – เป็นการยกเลิกการใช้ cd ครั้งล่าสุด

ตัวอย่างการใช้งาน :

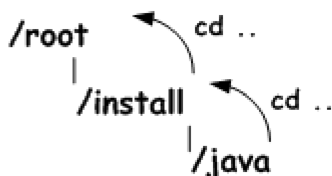
cd test

สั่งให้ย้ายไปยังไดเรคทรอรีชื่อ test

cd ../home/users/suchart

สั่งให้ย้ายไดเรคทรอรีขึ้นไป 1 ชั้นจากที่ทำงานอยู่ และจากนั้นจึงย้ายไปยังไดเรคทรอรี /home/users/suchart

cd ../../



สั่งย้ายไดเรคทรอรีขึ้นไปอีก 2 ระดับ เช่น เริ่มแรกอยู่ที่ไดเรคทรอรี java เมื่อ cd .. ครั้ง

แรกจะทำให้ย้ายขึ้นมาสู่ install และ cd .. ครั้งที่ 2 จะย้ายไปยัง root

ลองทดสอบ :

```
# cd
# cd ../..
```

ดูคำสั่งอื่นประกอบ : *pwd, chdir*

คำสั่ง pwd

File/Directory Basics : **pwd**	
คำสั่ง	**pwd** เป็นคำสั่งที่ใช้สำหรับแสดงไดเรคทรอรีปัจจุบันที่กำลังทำงานอยู่ (ในทำนองเดียวกับการพิมพ์ cd บน DOS) มาจากคำว่า print work directory
Syntax	pwd
Example	# pwd

ตัวอย่างการใช้งาน :

```
# pwd
/usr/local/bin
```
แสดงไดเรคทรอรีปัจจุบันที่กำลังทำงานอยู่(จากตัวอย่างกำลังทำงานอยู่ที่ /usr/local/bin)

ลองทดสอบ :

```
# cd /root; pwd;
# cd /usr/X11R6/bin/mkfont; pwd;
```

ดูคำสั่งอื่นประกอบ : *cd*

คำสั่ง mkdir

File/Directory Basics : **mkdir**	
คำสั่ง	**mkdir** เป็นคำสั่งที่ใช้สำหรับการสร้างไดเรคทรอรีใหม่ มาจากคำว่า make directory
Syntax	mkdir [*options*] *directories*
Example	# mkdir install

options :

-m *mode*, --mode *mode* สร้างไดเรคทรอรีพร้อมกับกำหนดสิทธิ์ในการใช้งานด้วย (ดูรายละเอียดจาก chmod)

-p, --parents สร้างไดเรคทรอรีพร้อมกับระบุ ไดเรคทรอรีในการสร้างได้

v, --verbose พิมพ์ผลลัพธ์ทุกครั้งที่มีการสร้างไดเรคทรอรี

--help แสดงการใช้คำสั่ง mkdir แบบย่อ

--version แสดงเวอร์ชันของคำสั่ง mkdir

ตัวอย่างการใช้งาน :

```
# mkdir mydir
```

สร้างไดเรคอทรอรีชื่อว่า mydir บนไดเรคทรอรีปัจจุบัน **# mkdir -m 444 personal** สร้างไดเรคอทรอรีชื่อว่า personal และมีสิทธิ์ในการเข้าถึงไฟล์ คือเจ้าของ สมาชิกในกลุ่ม และบุคคลอื่นๆ สามารถอ่านไฟล์นี้ได้อย่างเดียวเท่านั้น **# mkdir work; cd work mkdir junk; cd junk mkdir questions;** สร้างไดเรคอทรอรีชื่อว่า work, จากนั้นเข้าไปยังไดเรคอทรอรี work สร้างไดเรคทรอรีชื่อว่า junk เข้าไปยัง junk และสร้างไดเรคทรอรีชื่อว่า questions โดยการใช้ ; เพื่อบอกให้เชลล์ทำคำสั่งต่อกันไป ด้วยการสั่งงานเพียงครั้งเดียว **# mkdir -p work/junk/questions** สร้างไดเรคอทรอรีชื่อว่า questions โดยการระบุตำแหน่งที่ต้องการสร้าง(เมื่อไดเรคทรอรีไม่มี จะสร้างใหม่ให้อัตโนมัติ)

ลองทดสอบ :

```
# mkdir ../presentations
```

ดูคำสั่งอื่นประกอบ : *rmdir*

คำสั่ง rmdir

File/Directory Basics : **rmdir**	
คำสั่ง	**rmdir** เป็นคำสั่งที่ใช้สำหรับการลบไฟล์หรือไดเรคทรอรี ย่อมาจากคำว่า remove directory
Syntax	**rmdir [*options*] *directories***
Example	**# rmdir dir1** **# rmdir -rf /dir2**

options :

--ignore-fail-on-non-empty ทำการลบไดเรคทรอรี่โดยไม่สนใจความผิดพลาดที่เกิดจากการลบไดเรคทรอรีที่มีไฟล์ข้อมูลอยู่ภายใน

-p, --parents ทำการลบไดเรคทรอรีทั้งโครงสร้างออกทั้งหมด โดยเริ่มตั้งแต่ไดเรคทรอรีต้นทาง (parent) ไปจนถึงไดเรคทรอรีย่อยๆ (child) ทั้งหมด

-v, --verbose พิมพ์ผลลัพธ์ทุกครั้งที่มีการลบไดเรคทรอรี

--help แสดงการใช้คำสั่ง rmdir แบบย่อ

--version แสดงเวอร์ชันของคำสั่ง rmdir

ตัวอย่างการใช้งาน :

rmdir emptydir ลบไดเรคอทรอรีชื่อว่า emptydir โดยในไดเรคทรอรีดังกล่าวจะต้องเป็นไดเรคทรอรีว่าง

ตัวอย่างโครงสร้างไฟล์

```
#tree
|-- dir1
|  `-- dir2
|      `-- dir3
```

rmdir -p dir1

จากตัวอย่างโครงสร้างไฟล์ด้านบน เมื่อทำการลบไดเรคอทรอรีด้วยคำสั่ง rmdir –p /dir1 จะส่งผลให้ไดเรคทรอรีที่ซ่อนอยู่ภายในทั้งหมดถูกลบไปด้วย(ไดเรคทรอรีภายในต้องเป็นไดเรคทรอรีที่ว่างเท่านั้น ในกรณีที่มีไฟล์ข้อมูลอยู่จะไม่สามารถทำการลบออกได้)

rmdir --ignore-fail-on-non-empty dir1

ทำการลบไดเรคอทรอรี โดยไม่สนใจข้อผิดพลาดใดๆ

rmdir dir1/dir2/dir3 dir1/dir2 dir1

ทำการลบไดเรคทรอรีติดต่อกันหลายๆ ไดเรคทรอรีในครั้งเดียว จากตัวอย่าง เชลล์จะทำการลบไดเรคทรอรี dir3 ที่อยู่ใน dir1/dir2 ก่อน ต่อจากนั้นทำการลบ dir2 ที่อยู่ใน dir1 และสุดท้ายจะทำการลบ dir1 ซึ่งเป็นไดเรคทรอรีเริ่มต้นออกจากระบบ

ลองทดสอบ :

rm -r directory

ดูคำสั่งอื่นประกอบ : *mkdir, rm*

คำสั่ง tree

File/Directory Basics : **tree**	
คำสั่ง	**tree** เป็นคำสั่งที่ใช้แสดงรายละเอียดโครงสร้างไฟล์และไดเรคทรอรีแบบทรี ตามรูปแบบที่ต้องการ
Syntax	***tree [-adfgilnpqrstuxACDFNS] [-H baseHREF] [-T title] [-L level [-R]] [-P pattern] [-I pattern] [-o filename] [--version] [--help] [--inodes] [--device] [--noreport] [--nolinks] [--dirsfirst] [--charset charset] [<directory list>]***
Example	**# tree** **# tree -d /usr**

options :

--help ใช้แสดงความช่วยเหลือในการใช้งานคำสั่ง

--version แสดงเวอร์ชันของ tree

-a แสดงรายละเอียดโครงสร้างของไฟล์แบบทรีทั้งหมด ไม่รวมไฟล์แบบ ., ..

-d แสดงโครงสร้างทรีของไดเรคทรอรีเท่านั้น

-f แสดงที่อยู่ของไฟล์ในรูปแบบเต็ม (full path)

-i ไม่แสดงเส้นหรือสัญลักษณ์ในการเชื่อมโยงทรี

-q แสดงผลของอักษรที่ไม่สามารถแสดงผลให้เห็นได้ เป็นเครื่องหมาย ? แทน

-N แสดงสัญลักษณ์ ^ ด้วยอักษรที่ไม่สามารถแสดงผลได้

-p แสดง permission พร้อมกับโครงสร้างทรี

-u แสดงชื่อเจ้าของไฟล์หรือ UID

-g แสดงชื่อกลุ่มเจ้าของไฟล์หรือ GID

-s แสดงขนาดของไฟล์ มีหน่วยเป็นไบต์

-D แสดงวันที่ที่มีการแก้ไขไฟล์ล่าสุด

-F แสดงสัญลักษณ์ต่อท้ายเพื่อบอกความแตกต่าง เช่น / แสดงถึงไดเรคทรอรี, a คือ socket file, * คือไฟล์ที่ประมวลผลได้, | คือ FIFO

-r แสดงโครงสร้างทรีเป็นแบบสลับลำดับอักษร คือ z ไป a

-t แสดงโครงสร้างทรี โดยไฟล์ที่แก้ไขล่าสุดจะแสดงก่อน

-L level กำหนดการแสดงผลให้โครงสร้างทรีมีความลึกแค่ตัวเลข level

-A แสดงสัญลักษณ์ของเส้นที่สร้างทรีด้วยรหัส ASCII code

-S แสดงสัญลักษณ์แบบกราฟฟิกที่สร้างทรีด้วยรหัส ASCII code

-n ไม่แสดงผลเป็นแบบใช้สี

-C แสดงผลเป็นแบบใช้สี

-P pattern แสดงไฟล์ที่สอดคล้องกับ รูปแบบที่กำหนดใน pattern

-I pattern ไม่แสดงไฟล์ที่สอดคล้องกับ รูปแบบที่กำหนดใน pattern

-H baseHREF แสดงในรูปแบบของ HTML format

-T string เขียนทับข้อมูลในส่วนของ Header และ Title ของ HTML ด้วย string

-R เหมือน –L

-o file พิมพ์ผลลัพธ์ออกทางไฟล์ชื่อว่า file แทนการออกแทนจอภาพ

--inodes พิมพ์หมายเลข inode ของแต่ละไฟล์

--device พิมพ์หมายเลข device ที่กำกับอยู่กับทุกๆ ไฟล์

--noreport ไม่แสดงสรุปผลรวมของไฟล์และไดเรคทรอรี ในท้ายรายงาน

--nolinks ไม่แสดง Hyperlinks ในเอกสารประเภท HTML

--dirsfirst แสดงรายชื่อไดเรคทรอรีก่อนไฟล์

--charset X ใช้ตัวอักษร X ในการกำหนดเส้นทรี

ตัวอย่างการใช้งาน :

```
# tree
.
|-- Desktop
|   `-- wget-1.11.4-2.el5_4.1.i386.rpm
|-- sort.txt
```

```
|-- t1
|   |-- test.txt
|   `-- tx.txt
|-- t1.txt
|-- tar
|   |-- Desktop
|   |-- abc
|   |-- anaconda-ks.cfg
|   |-- install.log
|   |-- install.log.syslog
|   |-- sort.txt
19 directories, 56 files
```

แสดงโครงสร้างของไฟล์และไดเรคทรอรีทั้งหมดในรูปแบบโครงสร้างทรี

tree -L 1

```
|-- Desktop
|-- a
|-- abc
|-- abc.tar.gz
8 directories, 27 files
```

แสดงโครงสร้างทรีลึกแค่ 1 ลำดับเท่านั้น

```
# tree test
test
|-- a.txt
`-- t1
1 directory, 1 file
```

แสดงโครงสร้างทรีเฉพาะไดเรคทอรี test เท่านั้น

tree -d /usr

แสดงโครงสร้างทรีเฉพาะไดเรคทรอรี ใน /usr เท่านั้น

tree -d -L 2 /usr

แสดงโครงสร้างทรีเฉพาะไดเรคทรอรี ใน /usr เท่านั้น และทรีลึกเพียงแค่ 2 ระดับ

tree -dCA -L 2 /usr

แสดงโครงสร้างทรีเฉพาะไดเรคทรอรี ใน /usr เท่านั้น และทรีลึกเพียงแค่ 2 ระดับ และสัญลักษณ์เส้นของทรีเป็นสีแบบ ASCII code และเป็นเส้นทึบ

tree -d -L 1 / > tree-view.txt

แสดงโครงสร้างทรีเฉพาะไดเรคทรอรี ใน / เท่านั้น และทรีลึกเพียงแค่ 1 ระดับ ผลลัพธ์ที่ได้จะเก็บไว้ในไฟล์ชื่อว่า tree-view.txt

ลองทดสอบ :

```
# tree [-adfgilnpqrstuxACDFNS] [-H baseHREF] [-T title ] [-R]
   [-P pattern] [-I pattern] [-o filename] [--version] [--help] [--inodes]
   [--device] [--noreport] [--nolinks] [--dirsfirst] [--charset charset]
   [<directory list>]
```

หมายเหตุ : ควรทดลอง options อื่นๆ ให้ครบ

ดูคำสั่งอื่นประกอบ : *ls*

5.3.2 กลุ่มคำสั่งเกี่ยวกับ File Viewing

คำสั่ง cat

File Viewing : **cat**	
คำสั่ง	**cat** เป็นคำสั่งที่ใช้สำหรับดูข้อมูลภายในไฟล์ข้อมูล หรือ Standard Input และแสดงผลออกมาทาง Standard Output (เหมือนคำสั่ง type ใน Dos) ย่อมาจากคำว่า concatinat
Syntax	**cat [*options*] [*files*]**
Example	**# cat data.txt** **# cat file1.txt file2.txt > file3.txt**

options:-

-A, --show-all แสดงผลทุกๆ อักษร รวมทั้งตัวอักษรที่ไม่สามารถแสดงได้ เช่น อักขระที่ใช้ควบคุม ctrl ^d

-b, --number-nonblank นับจำนวนบรรทัด ไม่รวมบรรทัดที่ว่าง

-e เหมือน -vE

-E, --show-ends พิมพ์ $ ที่จุดจบของไฟล์

-n, --number นับจำนวนบรรทัด รวมบรรทัดว่างด้วย

-s, --squeeze-blank ถ้ามีบรรทัดว่างมากกว่า 1 บรรทัดต่อกัน จะลดให้เหลือเพียงบรรทัดเดียว

-t เหมือน -vT

-T, --show-tabs พิมพ์อักษร TAB ด้วยสัญลักษณ์ ^I แทน

-v, --show-nonprinting แสดงสัญลักษณ์ที่ไม่สามารถแสดงผลได้เช่น control character, LINEFEED, TAB เป็นต้น

ตัวอย่างการใช้งาน :

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
```

แสดงข้อมูลที่อยู่ในไฟล์รหัสผ่าน

```
# cat file1.txt file2.txt > file3.txt
```

นำข้อมูลใน file1.txt และ file2.txt มาเชื่อมต่อกัน จากนั้นเก็บไว้ใน file3.txt สัญลักษณ์ > เป็นการเปลี่ยนทิศทาง(redirection) การแสดงผลจากเดิมซึ่งออกทางจอภาพเปลี่ยนเป็นไฟล์แทน

```
# cat >> file1.txt
```

เป็นการเชื่อมต่อข้อมูล (append) ไปยังไฟล์ file1.txt โดยเมื่อใช้คำสั่งดังกล่าวเชลล์จะรอ

ให้ผู้ใช้งานป้อนข้อมูล เมื่อผู้ใช้ป้อนข้อมูลเรียบร้อยแล้วสามารถสั่งบันทึกด้วยการกดปุ่ม ctrl และปุ่ม d พร้อมกัน

```
# cat -t tabfil
```

แสดงข้อมูลในไฟล์ tabfil ถ้าข้อมูลในไฟล์ดังกล่าวมีตัวอักษร TAB จะทำการเปลี่ยนจาก TAB เป็นสัญลักษณ์ ^I แทน

```
# cat -n create-linux-file.txt
   1  this is my file
   2  Create file in linux using cat command
   3  my file name is create-linux-file.txt
   4  this is the line appends to create-linux-file.txt
   5  this is an example on using redirection to appends text
```

แสดงข้อมูลที่อยู่ในไฟล์ create-linux-file.txt โดยนับจำนวนบรรทัด รวมบรรทัดว่าง จำนวนบรรทัดจะแสดงตรงส่วนต้นของในแต่ละบรรทัด

```
# cat -v myfile.txt
```

แสดงข้อมูลที่อยู่ในไฟล์ myfile.txt โดยแสดงอักษรที่ไม่สามารถแสดงผลได้ทั้งหมด ยกเว้น line feed และ tab

```
# cat -b /etc/X11/XF86Config
```

แสดงข้อมูลที่อยู่ในไฟล์ XF86Config โดยนับจำนวนบรรทัด ไม่รวมบรรทัดว่าง

```
# cat -E /etc/X11/XF86Config
```

แสดงข้อมูลที่อยู่ในไฟล์ XF86Config โดยเติม $ ไว้ที่ท้ายไฟล์

ลองทดสอบ :

```
# cat file1.txt file2.txt | less
# cat foobar.txt | grep "barfoo"
# cat <file.txt >>file.txt
# cat my-long-file.txt | more
# cat -Tv myfile.txt
# cat RightNow.txt | wc
```

ดูคำสั่งอื่นประกอบ : *ed, pico, tac, tee, touch*

คำสั่ง more

File Viewing : **more**	
คำสั่ง	**more** เป็นคำสั่งที่ใช้สำหรับช่วยให้สามารถดูข้อมูลที่มีขนาดยาวๆ ได้ง่ายขึ้น โดยการแบ่งข้อมูลออกเป็นช่วงๆ ครั้งละ 1 หน้าจอภาพ
Syntax	**more [*options*] [*files*]**
Example	**# more myfile.txt** **# more +3 myfile.txt**

options:-

+num ระบุบรรทัดเริ่มต้นที่ต้องการแสดงผล

-num *number* ระบุจำนวนบรรทัด สำหรับการแสดงผลใน 1 หน้าจอภาพ

+/*pattern* แสดงผลบรรทัดแรกตามรูปแบบที่กำหนดใน pattern

-c เคลียร์หน้าจอภาพก่อนแสดงผล

ภายในโปรแกรม more จะมีคำสั่งเพื่อใช้งานคราวๆ ดังนี้

= แสดงเลขบรรทัด

q ออกจากโปรแกรม

<space> เลื่อนไปยังหน้าถัดไป

<enter> เลื่อนไปยังบรรทัดถัดไป

ตัวอย่างการใช้งาน :

more data.txt
ดูไฟล์ข้อมูล data.txt ครั้งละ 1 หน้าจอ
more -c index.php
เคลียร์หน้าจอก่อนแสดงผลข้อมูลในไฟล์ index.php
more +3 myfile.txt
แสดงข้อมูลในไฟล์ชื่อว่า myfile.txt ทีละ 1 จอภาพ โดยเริ่มต้นแสดงข้อมูลที่บรรทัดที่ 3 ของ myfile.txt เป็นบรรทัดแรกบนหน้าจอภาพ

ลองทดสอบ :

```
# ls -al | more
# cat /var/log/messages | more
```

ดูคำสั่งอื่นประกอบ : *cat, csh, ctags, less, man, nroff, script, sh, ul*

คำสั่ง less

File Viewing : **less**	
คำสั่ง	**less** เป็นคำสั่งที่ใช้สำหรับดูข้อมูลในไฟล์ทีละหน้าจอ เหมือนคำสั่ง more แต่พัฒนาความสามารถให้สูงขึ้น โดยสามารถย้อนดูข้อมูลที่ผ่านมาได้ และคุณสมบัติอื่นๆ เพิ่มเติมอีกหลายอย่าง
Syntax	**less [*options*] [*filename*]**
Example	**# less file.txt**

options:-

+n เริ่มต้นแสดงข้อมูลที่บรรทัดที่ n ในไฟล์ข้อมูล

-c เครียร์หน้าจอภาพก่อนแสดงผล

h H แสดงข้อความช่วยเหลือ

q :q Q :Q ZZ ออกจาก less

คำสั่งเคลื่อนย้าย

e, ^E, j, ^N, CR เลื่อนบรรทัดไปท้ายไฟล์ทีละ 1 บรรทัด(^ คือกดปุ่ม Ctrl)

y, ^Y, k, ^K, ^P เลื่อนบรรทัดไปต้นไฟล์ทีละ 1 บรรทัด(กดปุ่ม Ctrl พร้อมกับ Y)

f, ^F, ^V, SPACE เลื่อนข้อมูลไปท้ายไฟล์ทีละ 1 จอภาพ

b, ^B, ESC-v เลื่อนข้อมูลไปต้นไฟล์ทีละ 1 จอภาพ

d, ^D เลื่อนข้อมูลไปท้ายไฟล์ทีละครึ่งจอภาพ

u, ^U เลื่อนข้อมูลไปต้นไฟล์ทีละครึ่งจอภาพ

n ค้นหาข้อมูลซ้ำอีกครั้งจากต้นไฟล์ไปท้ายไฟล์

N ค้นหาข้อมูลซ้ำอีกครั้งจากท้ายไฟล์ไปต้นไฟล์

ค้นหา

/pattern ค้นหาคำตามที่กำหนดใน pattern ค้นหาจากต้นไฟล์ไปท้ายไฟล์

?pattern ค้นหาคำตามที่กำหนดใน pattern ค้นหาจากท้ายไฟล์ไปต้นไฟล์

ตัวอย่างการใช้งาน :

```
# less +3 index.php
```
แสดงข้อมูลในไฟล์ index.php ตั้งแต่บรรทัดที่ 3 ขึ้นไป ทีหน้า 1 หน้าจอภาพ

```
# less file.txt
```
แสดงข้อมูลในไฟล์ file.txt ครั้งละ 1 จอภาพ

```
# ls | less
```
แสดงข้อมูลจากผลที่ได้จากคำสั่ง ls ครั้งละ 1 จอภาพ

```
# cat file.txt | less
```
แสดงข้อมูลจากผลที่ได้จากคำสั่ง cat ครั้งละ 1 จอภาพ

ลองทดสอบ :

```
# ls -l *.conf | less
# less file1 file2
```

ดูคำสั่งอื่นประกอบ : *more*

คำสั่ง head

File Viewing : **head**	
คำสั่ง	head เป็นคำสั่งที่ใช้สำหรับแสดงส่วนหัวของไฟล์ข้อมูล ตามจำนวนบรรทัดที่ต้องการ
Syntax	head [*options*] [*files*]
Example	# head myfile.txt # head -15 myfile.txt

options:-

-c *num*[b|k|m] , --bytes *num* [b|k|m] แสดงขนาดของข้อมูลตามที่กำหนด เมื่อกำหนดเป็น b จะแสดงผลเป็นไบต์, k = กิโลไบต์, m= เมกกะไบต์

--help แสดงคำสั่งช่วยเหลือ

-n *num*, --lines *num*, -num แสดงผลตามจำนวนบรรทัดที่กำหนด ถ้าไม่กำหนดไว้ค่าเริ่มต้นจะเท่ากับ 10 บรรทัด

-q, --quiet, --silent ไม่แสดง header ของไฟล์

-v, --verbose แสดง header ของไฟล์

--version แสดงเวอร์ชันของ head

ตัวอย่างการใช้งาน :

head myfile.txt

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ myfile.txt โดยแสดงข้อมูล 10 บรรทัดเท่านั้น(ค่า default)

head -15 myfile.txt

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ myfile.txt โดยแสดงข้อมูล 15 บรรทัดตามจำนวนที่ระบุไว้

head –n 20 myphone_number.txt

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ myphone_number.txt โดยแสดงข้อมูล 20 บรรทัดเท่านั้น

head -2 install.log.syslog number.txt

```
==> install.log.syslog <==
<86>Nov  8 15:54:25 groupadd[858]: new group: name=rpm, GID=37
<86>Nov  8 15:54:26 useradd[862]: new user: name=rpm, UID=37, GID=37,
home=/var/lib/rpm, shell=/sbin/nologin

==> number.txt <==
1 6 3 8 3 5 6 0
1 6 3 8 3 5 6 0
```

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ install.log.syslog และ number.txt พร้อมกัน โดยแสดงข้อมูลของ install.log.syslog ก่อน 2 บรรทัดแรก จากนั้นจะแสดงข้อมูลของไฟล์ number.txt อีก 2 บรรทัด

head –c 5 install.log.syslog

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ install.log.syslog เฉพาะ 5 ไบต์แรกเท่านั้น

head -c1k install.log.syslog

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ install.log.syslog ไม่เกิน 1 กิโลไบต์ไบต์

head –c1m install.log.syslog

แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ install.log.syslog ไม่เกิน 1 เม็กกะไบต์

ls | head

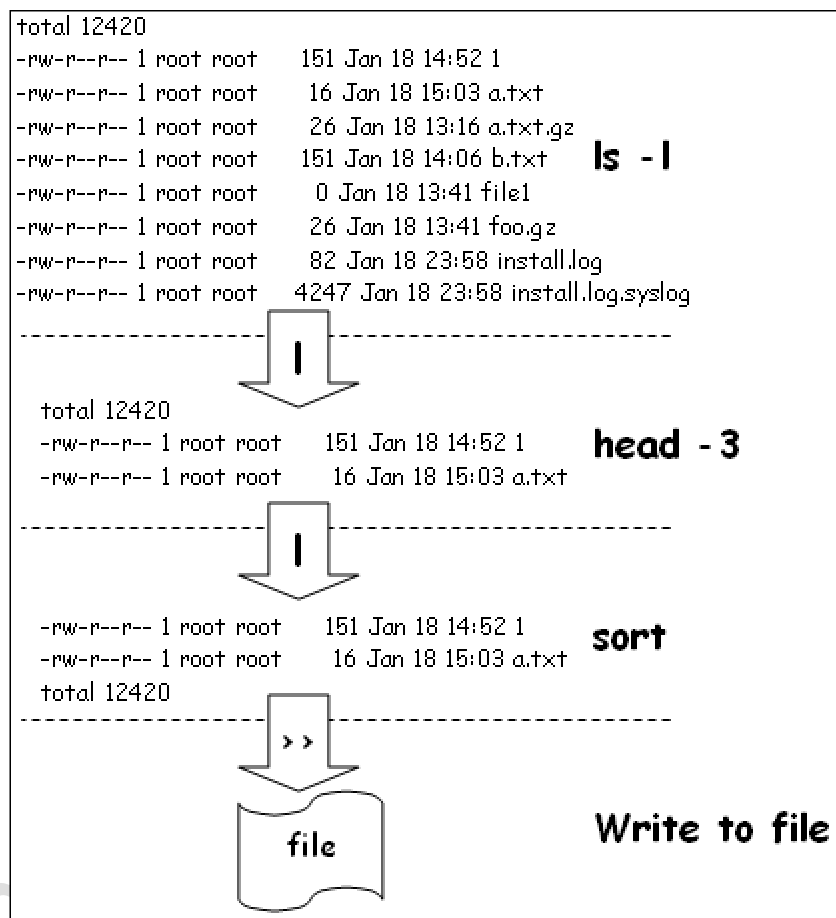
แสดงข้อมูลที่ได้รับจากคำสั่ง ls ผ่านทางไปป์ เฉพาะ 10 บรรทัดแรก

ls -l | head -3 | sort >> file

ผลลัพธ์ที่ได้จากคำสั่ง ls –l จะส่งต่อให้กับคำสั่ง head เพื่อแสดงส่วนหัวของไฟล์ 3 อันดับแรก จากนั้นผลลัพธ์ที่ได้จะถูกส่งต่อไปยัง sort ให้จัดเรียงตามลำดับตัวอักษร สุดท้ายจะใช้การเปลี่ยนทิศทาง (>>) นำผลลัพธ์ไปเก็บไว้ในไฟล์(แสดงในรูปที่ 5-6)

head -5 *

แสดงผลลัพธ์ 5 บรรทัดของทุกๆ ไฟล์ในไดเรคทรอรีปัจจุบัน



รูปที่ 5-6 แสดงการใช้คำสั่ง ls –l | head -3 | sort >> file

ลองทดสอบ :

```
# head -n 12 file1 > file2
# head -n 12 file1 >> file2
# grep '(202)' phone_list | head
# head -99999 file1 file2 file3
```

ดูคำสั่งอื่นประกอบ : *cat, more, pg, tail*

คำสั่ง tail

File Viewing : **tail**

คำสั่ง	**tail** เป็นคำสั่งที่ใช้สำหรับแสดงข้อมูลในส่วนท้ายของไฟล์ข้อมูล ตามจำนวนบรรทัดที่ต้องการ
Syntax	**tail [*options*] [*files*]**
Example	**# tail myfile.txt** **# tail myfile.txt -n 100**

options:-

-c *num*, --bytes *num* แสดงข้อมูลท้ายไฟล์ มีจำนวนเท่ากับ num ไบต์

-f, --follow[=name|descriptor] แสดงผลท้ายไฟล์ โดยแสดงแบบต่อเนื่อง ถ้ามีข้อมูลเพิ่มเข้ามาในไฟล์ดังกล่าว จะแสดงผลไปเรื่อยๆ จนกว่าจะหยุดด้วยการใช้คำสั่ง q หรือ Ctrl + C

-n *num*, --lines=*num* แสดงข้อมูลท้ายไฟล์ตามจำนวนบรรทัดที่กำหนด

-s *sec*, --sleep-interval=*sec* ใช้กับการแสดงผลกับ –f โดย option ดังกล่าวจะแสดงผลและหยุดแสดงผลสลับกันไปเรื่อยๆ ตามเวลาที่กำหนด

b แสดงผลเป็นจำนวนเท่ากับไบต์

k แสดงผลเป็นจำนวนเท่ากับกิโลไบต์

m แสดงผลเป็นจำนวนเท่ากับเมกกะไบต์

ตัวอย่างการใช้งาน :

tail myfile.txt
แสดงข้อมูลในส่วนท้ายของไฟล์ (ค่า default คือ 10 บรรทัด)
tail myfile.txt -n 100
แสดงข้อมูลในส่วนท้ายของไฟล์ จำนวน 100 บรรทัด
grep '\.Ah' myfile | tail -20
ค้นหาข้อความจากไฟล์ myfile ที่มีข้อความคือ .Ah ผลลัพธ์ที่ได้จาก grep จะส่งไปยัง tail ให้แสดผลส่วนท้ายของข้อมูลเพียง 20 บรรทัดเท่านั้น
tail -2b bigfile
แสดงข้อมูลไฟล์ที่มีขนาดใหญ่มาก โดยแสดงแค่ 1,024 ไบต์เท่านั้น(2 x 512)
tail --lines=5 /var/log/messages
ดูข้อมูลส่วนท้ายของไฟล์ /var/log/messages เท่ากับ 5 บรรทัด
tail --lines=150 /var/log/html/access_log > access_backup_log
ดูข้อมูลส่วนท้ายของไฟล์ /var/log/messages เท่ากับ 150 บรรทัด แล้วส่งไปเก็บไว้ในไฟล์ชื่อว่า access_backup_log (ในกรณีที่ต้องการสำรองข้อมูล)
tail -c 5 index.php
แสดงข้อมูลในส่วนท้ายของไฟล์เท่ากับ 5 ไบต์
tail –f /var/log/access.log
แสดงข้อมูลในส่วนท้ายของไฟล์ access.log อย่างต่อเนื่องแบบ realtime

ลองทดสอบ :

tail +50 file.txt

```
# tail -f /var/log/messages
# tail -f error_log
# alias mess='tail -100 /var/log/messages  | more'
# ls | tail >> last_filenames
# ls | tail | sort -r >> last_filenames
```

ดูคำสั่งอื่นประกอบ : *cat, head, more, pg*

คำสั่ง nl

File Viewing : **nl**	
คำสั่ง	**nl** เป็นคำสั่งที่ใช้สำหรับแสดงข้อมูลพร้อมกับหมายเลขบรรทัดในไฟล์ข้อมูล ตามเงื่อนไขที่ระบุไว้ในออพชัน
Syntax	**nl [*options*] [*files*]**
Example	**# nl myfile.txt**

options:-

-b p^pattern แสดงหมายเลขบรรทัด ในไฟล์ข้อมูล เฉพาะบรรทัดที่ระบุในตัวแปร pattern เท่านั้น

ตัวอย่างการใช้งาน :

```
# nl myfile.txt
1  What Vim Can Do
2  Vim is an advanced text editor that seeks to provide the
3  power of the de-facto Unix editor 'Vi', with a more
4  complete feature set. It's useful whether you're already
5  using vi or using a different editor. Users of Vim 5 and 6
6  should consider upgrading to Vim 7. The main advantages
7  of Vim 6 compared to Vim 5 can be found on this page.
```

แสดงข้อมูลในไฟล์ myfile.txt โดยมีหมายเลขบรรทัดระบุ ในส่วนต้นของบรรทัด

```
# nl myfile.txt -b p^power
     What Vim Can Do
     Vim is an advanced text editor that seeks to provide the
  1  power of the de-facto Unix editor 'Vi', with a more
     complete feature set. It's useful whether you're already
     using vi or using a different editor. Users of Vim 5 and 6
     should consider upgrading to Vim 7. The main advantages
     of Vim 6 compared to Vim 5 can be found on this page.
```

แสดงข้อมูลพร้อมหมายเลขบรรทัดในส่วนต้นของบรรทัด เฉพาะบรรทัดที่มีคำว่า power เท่านั้น

ดูคำสั่งอื่นประกอบ : *more, less, ls, cat*

คำสั่ง od [32]

File Viewing : **od**	
คำสั่ง	**od** เป็นคำสั่งที่ใช้สำหรับแสดงข้อมูลของไฟล์ในรูปแบบของไบรารี
Syntax	od [*options*] [*files*]
Example	# od myfile.txt # od --format=c myfile.txt

options:-

-A, --address-radix=radix กำหนดรูปแบบของข้อมูลที่ต้องการแสดงผล โดยกำหนดที่ radix มีรูปแบบให้เลือกใช้ดังต่อไปนี้

- d กำหนดให้แสดงผลเป็นเลขฐานสิบ
- o กำหนดให้แสดงผลเป็นเลขฐานแปด
- x กำหนดให้แสดงผลเป็นเลขฐานสิบหก
- n ยกเลิกรูปแบบ

-N, --read-bytes=BYTES กำหนดขนาดของข้อมูลในการแสดงผล โดยระบุข้อมูลที่ต้องการแสดงผลมีหน่วยเป็นไบต์

-t, --format=TYPE กำหนดรูปแบบในการแสดงผลแบบละเอียด โดยสามารถกำหนดในตัวแปร type ดังนี้

- a แสดงผลในรูปแบบของตัวอักษร
- c แสดงเป็นรหัส ascii ร่วมกับ \
- d[SIZE] กำหนดขนาดแสดงผลของเลขฐานสิบ(มีขนาดเป็นไบต์ต่อ 1 ตัวเลข)
- f[SIZE] กำหนดขนาดแสดงผลของเลขจำนวนจริง(มีขนาดเป็นไบต์ต่อ 1 ตัวเลข)
- o[SIZE] กำหนดขนาดแสดงผลของเลขฐานแปด(มีขนาดเป็นไบต์ต่อ 1 ตัวเลข)
- u[SIZE] กำหนดขนาดแสดงผลของเลขฐานสิบแบบไม่คิดเครื่องหมาย(มีขนาดเป็นไบต์ต่อ 1 ตัวเลข)
- x[SIZE] กำหนดขนาดแสดงผลของเลขฐานสิบหก(มีขนาดเป็นไบต์ต่อ 1 ตัวเลข)

-w, --width[=BYTES] กำหนดขนาดการแสดงผลในแต่ละบรรทัดมีความกว้างเท่ากับ BYTES

-a แสดงข้อมูลเป็นตัวอักษร (เหมือนกับใช้ออพชัน –format=a)

-b แสดงข้อมูลเป็นไบต์ (เหมือนกับใช้ออพชัน –format=oc)

-c แสดงข้อมูลเป็น ascii (เหมือนกับใช้ออพชัน –format=c)

-d แสดงข้อมูลเป็นฐานสิบโดยไม่คิดเครื่องหมาย (เหมือนกับใช้ออพชัน –format=u)

-x แสดงข้อมูลเป็นฐานสิบหก (เหมือนกับใช้ออพชัน –format=x)

ตัวอย่างการใช้งาน :

ตัวอย่างไฟล์ข้อมูล myfile.txt

```
Vim is an advanced text editor that seeks to provide the
power of the de-facto Unix editor 'Vi', with a more
complete feature set. It's useful whether you're already
# od myfile.txt
0000000 064526 020155 071551 060440 020156 062141 060566 061556
0000020 062145 072040 074145 020164 062145 072151 071157 072040
0000040 060550 020164 062563 065545 020163 067564 070040 067562
```

แสดงข้อมูลในไฟล์ myfile.txt เป็นเลขฐานแปด(ค่า default ของ od) ในตัวอย่างคอลัมภ์แรก คือ 0000000 เป็นตัวเลขแสดงจำนวนอักษรในแถวแรก โดยปกติจะแสดงแถวละ 20 ตัวอักษร และแถวที่สองจะเริ่มตั้งแต่ตัวอักษรที่ 20 ตามลำดับ ในแถวแรกอักษรตัวแรกคือ 064 คือตัวอักษร V และตัวสุดท้ายของแถวคือ 556 คือตัว c

```
# od --format=c myfile.txt
0000000  V  i  m     i  s     a  n     a  d  v  a  n  c
0000020  e  d     t  e  x  t     e  d  i  t  o  r     t
0000040  h  a  t     s  e  e  k  s     t  o     p  r  o
0000060  v  i  d  e     t  h  e \n  p  o  w  e  r     o
```

แสดงข้อมูลในไฟล์ myfile.txt โดยกำหนดรูปแบบการแสดงผลเป็นรหัส ascii แทนการแสดงเป็นเลขฐานแปด

```
# od --address-radix=x myfile.txt
```

แสดงข้อมูลในไฟล์ myfile.txt โดยกำหนดรูปแบบการแสดงผลเป็นเป็นเลขฐานสิบหก

```
# od --format=f 2 myfile.txt
0000000  1.479317822374409e-259  1.642552904128327e-259
0000020  2.316339989845287e-152  5.049779884467015e+223
```

แสดงข้อมูลในไฟล์ myfile.txt โดยกำหนดรูปแบบการแสดงผลเป็นเป็นเลขจำนวนจริง ในตัวเลขจำนวนจริงแต่ละค่าจะมีค่าได้ไม่เกิน 2 ไบต์

```
# od -N 16 -c /bin/sh
0000000 177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0  \0
```

แสดงข้อมูลในไฟล์เชลล์ sh โดยแสดง 16 ตัวอักษรแรกเป็นรหัส ascii หรือ backslash escapes ในรูปฐานแปด

```
# od -N 16 -a /bin/sh
0000000 del   E   L   F soh soh soh nul nul nul nul nul nul nul nul nul
```

แสดงข้อมูลในไฟล์เชลล์ sh โดยแสดง 16 ตัวอักษรแรกเป็นตัวอักษร

```
# od -N 16 -t x1 /bin/sh
0000000 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
```

แสดงข้อมูลในไฟล์เชลล์ sh โดยแสดง 16 ตัวอักษรแรกเป็นตัวเลขฐานสิบหกแบบสั้น

ลองทดสอบ :

```
# dir | od -c | more
# cat my_file | od -c |more
# od my_file |more
```

```
# od -N 16 /bin/sh
```

ดูคำสั่งอื่นประกอบ : *cat, head, more*

5.3.3 กลุ่มคำสั่งเกี่ยวกับ File Creation and Editing

คำสั่ง vim, vi

เป็นโปรแกรมที่ใช้สำหรับสร้างและแก้ไขไฟล์ข้อความ สามารถอ่านเพิ่มเติมได้ในบทที่ 6

คำสั่ง umask

File Creation and Editing : **umask**	
คำสั่ง	**umask** เป็นคำสั่งที่ใช้สำหรับแสดงหรือกำหนดค่าเริ่มต้นของไฟล์ข้อมูล
Syntax	**od [*options*] [*files*]**
Example	**# umask** **# od --format=c myfile.txt**

umask (user file-creation mode mask) เป็นเลขฐานแปดจำนวน 12 บิต(4 ชุด ชุดละ 3 บิต)ใช้สำหรับระบุค่าสิทธิ์ในการใช้งานไฟล์ที่สร้างใหม่ในระบบ ลินุกซ์สามารถตรวจสอบค่า umask ได้โดยใช้คำสั่ง umask (ค่า default คือ 0022)



หมายเหตุ: สิทธิ์ของไฟล์ที่เคอร์เนลสร้างขึ้นในขณะทำการติดตั้งเริ่มต้นคือ

ไฟล์ข้อมูลธรรมดา (text file) = 0666

ไฟล์ที่สามารถประมวลผลได้ (Executable) = 0777

การใช้งาน umask สามารถทำได้โดยการนำค่าเลขฐานแปดที่ตามหลังคำสั่ง umask นำไปทำการ XOR กับค่าดีฟอลต์ในการสร้างไฟล์ในระบบ ซึ่งเป็นค่าที่ถูกกำหนดขึ้นอัตโนมัติในขณะติดตั้งระบบปฏิบัติการ โดยมีค่าดังตาราง

ประเภทของไฟล์	**ค่าเริ่มต้นของไฟล์ที่ระบบสร้างขึ้น**
ไฟล์ข้อมูลธรรมดา (text file)	0666
ไฟล์ที่สามารถประมวลผลได้ (Executable)	0777

ตัวอย่างที่ 1 ถ้าผู้ใช้งานกำหนดค่า umask เป็น 0022 ดังนั้นไฟล์ที่สร้างขึ้นใหม่จะไม่มีความสามารถในการประมวลผลได้ จะมีสิทธิ์บนระบบไฟล์คือ 0644 หรือ -rw-r--r—ดังตารางข้างล่าง

ประเภทของไฟล์	**ค่าเริ่มต้น(เลขฐาน 8)**	**ค่าเริ่มต้น(เลขฐาน 2)**
ค่าเริ่มต้นของไฟล์ที่ประมวลผลได้	0666	000 110 110 110
ค่า umask ที่กำหนด	0022	000 000 010 010

ผลลัพธ์ที่ได้จากการ XOR ระหว่างค่าเริ่มต้นของไฟล์ข้อมูลธรรมดาที่ระบบสร้างกับค่า umask ที่กำหนด	0666 ⊕ 0022	000 110 110 110 ⊕ 000 000 010 010 **<u>000 110 100 100</u>** **0 6 4 4**

ตัวอย่างที่ 2 ถ้าผู้ใช้งานกำหนดค่า umask เป็น 0022 ดังนั้นไฟล์ที่สร้างขึ้นใหม่จะมีความสามารถในการประมวลผลได้ จะมีสิทธิ์บนระบบไฟล์คือ 0755 หรือ -rwxr-xr-x ดังตารางข้างล่าง

ประเภทของไฟล์	**ค่าเริ่มต้น(เลขฐาน 8)**	**ค่าเริ่มต้น(เลขฐาน 2)**
ค่าเริ่มต้นของไฟล์ที่ประมวลผลได้	0777	000 111 111 111
ค่า umask ที่กำหนด	0022	000 000 010 010
ผลลัพธ์ที่ได้จากการ XOR ระหว่างค่าเริ่มต้นของไฟล์ข้อมูลธรรมดาที่ระบบสร้างกับค่า umask ที่กำหนด	0666 ⊕ 0022	000 111 111 111 ⊕ 000 000 010 010 **<u>000 111 101 101</u>** **0 7 5 5**

ตารางความจริงของ XOR

Input		**Output**
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

ตัวอย่างการใช้งาน :

umask 000

กำหนดคุณสมบัติ เริ่มต้นให้กับไฟล์ โดยเมื่อกำหนดเป็น 000 แสดงว่า ในอนาคตจะอนุญาติให้สามารถกำหนดสิทธิ์บนไฟล์ดังกล่าวได้เต็มรูปแบบตามความต้องการ

umask 0077

กำหนดคุณสมบัติ เริ่มต้นให้กับไฟล์ เมื่อเป็นไฟล์แบบธรรมดาจะได้ผลลัพธ์คือ 0077 XOR 0677 = 0600 เจ้าของไฟล์มีสิทธิ์ในการเขียนและอ่านได้ แต่สมาชิกในกลุ่มและกลุ่มอื่นๆ ไม่มีสิทธ์ใดๆ กับไฟล์ดังกล่าว

umask 0022

ลองทดสอบ :

```
# umask 0111
# umask 0222
# umask 0333
```

ดูคำสั่งอื่นประกอบ : *chmod, csh, ksh, sh*

5.3.4 กลุ่มคำสั่งเกี่ยวกับ File Properties

คำสั่ง stat

File Properties : **stat**	
คำสั่ง	**stat** เป็นคำสั่งที่ใช้สำหรับแสดงสถานะ สถิติ หรือคุณลักษณะของไฟล์ข้อมูล
Syntax	**stat [*options*] [*files*]**
Example	**# stat myfile.txt** **# stat index.html**

options:-

-L, --dereference รายงานคุณสมบัติของไฟล์ พร้อมกับลิงค์ที่เชื่อมโยง

-Z, --context รายงานข้อมูลเกี่ยวกับการรักษาความปลอดภัยของไฟล์ คำสั่งนี้จะต้องใช้ร่วมกับ SELinux ด้วย

-f, --file-system รายงานคุณสมบัติ ชนิด และโครงสร้างของไฟล์

-t, --terse รายงานคุณสมบัติของไฟล์แบบสั้น

ตัวอย่างการใช้งาน :

```
# stat test
File: `test'
  Size: 212           Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 1016395      Links: 1
Access: (0777/-rwxrwxrwx)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2010-02-03 03:23:33.000000000 +0700
Modify: 2010-01-18 15:49:25.000000000 +0700
Change: 2010-02-03 23:19:26.000000000 +0700
```

รายงานคุณสมบัติของไฟล์ test จากตัวอย่างคำสั่ง stat จะรายงานคุณสมบัติต่างๆ ดังนี้ ขนาดของไฟล์ (212 ไบต์) เป็นชนิด text ไฟล์(regular) มีหมายเลข Inode คือ 1016395 มีจำนวนลิงค์เชื่อมโยง 1 ลิงค์ เวลาการเข้าถึงล่าสุดคือ 2010-02-03 เจ้าของคือ root และไฟล์ดังกล่าวมีสิทธิ์คือ 0777

```
# stat -f test
File: "test"
    ID: 0        Namelen: 255     Type: ext2/ext3
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 1745792    Free: 1102826    Available: 1012714
```

```
Inodes: Total: 1802240    Free: 1708404
```
รายงานคุณสมบัติของไฟล์ test จากตัวอย่างจะแสดงชนิดของไฟล์แบบ ext2/ext3

```
# stat -L test
File: `test'
  Size: 212          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 1016395     Links: 1
Access: (0777/-rwxrwxrwx)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2010-02-03 03:23:33.000000000 +0700
Modify: 2010-01-18 15:49:25.000000000 +0700
Change: 2010-02-03 23:19:26.000000000 +0700
```
รายงานคุณสมบัติของไฟล์ test จากตัวอย่างจะแสดงจำนวนลิงค์เชื่อมโยง

ลองทดสอบ :

```
# stat /
# stat -f /
```

ดูคำสั่งอื่นประกอบ : *cat, head, more*

คำสั่ง wc

File Properties : **WC**	
คำสั่ง	**wc** เป็นคำสั่งที่ใช้สำหรับนับจำนวนคำอักษร จำนวนบรรทัด และจำนวนคำในไฟล์ข้อมูล
Syntax	**wc [*options*] [*files*]**
Example	**# wc myfile.txt** **# wc -l myfile.txt**

options:-

-c, --bytes นับจำนวนไบต์

-m, --chars นับจำนวนตัวอักษร

-l, --lines นับจำนวนแถว

-L, --max-line-length นับจำนวนตัวอักษร เฉพาะแถวที่มีความยาวที่สุดในไฟล์

-w, --words นับจำนวนคำ (แต่ละคำจะถูกแยกด้วยช่องว่าง)

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

```
# wc /etc/passwd
38   56 1675 /etc/passwd
```
นับจำนวนคำในไฟล์ข้อมูล /etc/passwd ผลลัพธ์ที่ได้คือ นับจำนวนบรรทัดได้ 38 บรรทัด 56 คำ และ 1675 ตัวอักษร

```
# wc -l /etc/passwd
38 /etc/passwd
```

นับเฉพาะจำนวนบรรทัดในไฟล์ /etc/passwd ได้ 38 บรรทัด

```
# wc -w /etc/passwd
56 /etc/passwd
```

นับเฉพาะจำนวนคำในไฟล์ /etc/passwd ได้ 56 คำ

```
# wc -m /etc/passwd
1675 /etc/passwd
```

นับเฉพาะจำนวนตัวอักษรในไฟล์ /etc/passwd ได้ 1675 คำ

```
# wc -L /etc/passwd
69 /etc/passwd
```

นับเฉพาะจำนวนตัวอักษรแถวที่ยาวที่สุดในไฟล์ /etc/passwd ได้ 69 ตัวอักษร

```
# wc file1 file2 file3
8  68 357 file1
2   2  16 file2
5   4  17 file3
15  74 390 total
```

นับจำนวนแถว คำ และตัวอักษร ในไฟล์ข้อมูล file1, file2 และ file3 พร้อมกันในคำสั่งเดียว ผลที่ได้คือจะแสดงข้อมูลของแต่ละไฟล์แยกกันก่อน จากนั้นจะแสดงผลรวมของไฟล์ทั้งสามอีกครั้งในบันทัดสุดท้าย

ลองทดสอบ :

```
# who | wc -l
# ps -e | wc -l
# wc . *
# wc -lw file5
# ls | wc -l
# ls -l | grep ^d | wc -l
# dir | wc
# cat my_file | wc
```

ดูคำสั่งอื่นประกอบ : *cksum*

คำสั่ง file

File Properties : **file**	
คำสั่ง	file เป็นคำสั่งที่ใช้สำหรับตรวจสอบและแสดงประเภทของไฟล์ข้อมูล
Syntax	**file [*options*] [*files*]**
Example	**# file myfile.txt** **# file ***

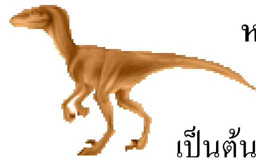
options:-

-c, --checking-printout ตรวจสอบไฟล์ข้อมูลและพิมพ์รายงานเมื่อไฟล์ที่ตรวจสอบเกิดความผิดพลาด ซึ่งรูปแบบของไฟล์ในลินุกซ์ที่ใช้ตรวจสอบจะเก็บอยู่ที่ /usr/share/magic

-m file ตรวจสอบไฟล์ข้อมูลด้วยรูปแบบของไฟล์ที่ระบุใน file แทนระบบไฟล์ที่ลินุกซ์ใช้ตรวจสอบแบบปกติ (เก็บอยู่ที่ /usr/share/magic)

-f file อ่านไฟล์ข้อมูลที่ระบุใน file ไปทำการตรวจสอบ

--help คำสั่งช่วยเหลือ



หมายเหตุ: บนระบบปฏิบัติการ ดอสหรือวินโดวส์นั้น ประเภทของไฟล์ข้อมูลจะถูกระบุด้วยนามสกุล เช่น .exe, .doc, .txt เป็นต้น แต่ในยูนิกซ์/ลินุกซ์ จะไม่มีนามสกุลเพื่อใช้สำหรับระบุประเภทของไฟล์ข้อมูล ดังนั้นการหาประเภทของไฟล์ข้อมูล ยูนิกซ์จะทำการตรวจสอบเนื้อหาภายในของไฟล์ ซึ่งคำสั่ง file จะทำการหน้าที่ดังกล่าว

ตัวอย่างการใช้งาน :

file myfile.txt
myfile.txt: ASCII English text
แสดงประเภทของ myfile.txt เป็น text ไฟล์ธรรมดา

file /bin/sh
/bin/sh: symbolic link to `bash'
แสดงประเภทของไฟล์ /bin/sh ซึ่งผลจากการตรวจสอบเป็นประเภทลิงค์ไฟล์ ที่เชื่อมโยงไปยังไฟล์อีกไฟล์หนึ่งคือ /bin/bash

file /bin/bash
/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.9, dynamically linked (uses shared libs), for GNU/Linux 2.6.9, stripped
ไฟล์ /bin/bash เป็นไฟล์ประเภทประมวลผลได้ (executable) และสามารถทำงานได้บนระบบปฏิบัติการลินุกซ์

file *
shutdown.htm: HTML document text
si.htm: HTML document text
side0.gif: GIF image data, version 89a, 107 x 18
robots.txt: ASCII text, with CRLF line terminators
routehlp.htm: HTML document text
rss: setgid directory
ตรวจสอบชนิดของไฟล์ในไดเรคทรอรีปัจจุบัน ซึ่งประกอบไปด้วย ไฟล์ shutdown.html, si.html, routehlp.htm เป็นไฟล์ text, side0.gif เป็นไฟล์รูปภาพชนิด gif, robots.txt เป็นไฟล์ ascii และ rss เป็นไดเรคทรอรี

file *.txt
form.txt: news or mail text
friend.txt: news or mail text
index.txt: ASCII Java program text, with very long lines, with CRLF line terminators

```
news.txt: Non-ISO extended-ASCII C program text, with very long lines, with CRLF line terminators
refrence.txt: news or mail text
robots.txt: ASCII text, with CRLF line terminators
stopwords.txt: ASCII English text, with CRLF line terminators
```

ตรวจสอบชนิดของไฟล์ในไดเรคทรอรีปัจจุบัน ซึ่งจะรายงานเฉพาะไฟล์ที่มีส่วนขยายเป็น .txt เท่านั้น

ลองทดสอบ :

```
# file -i *.txt
```

ดูคำสั่งอื่นประกอบ : *ls, wc*

คำสั่ง touch

File Properties : **touch**	
คำสั่ง	**touch** เป็นคำสั่งที่ใช้สร้างไฟล์ว่าง(empty file)และเปลี่ยนค่าเวลาของไฟล์ (โดยปกติเวลาของไฟล์ในยูนิกซ์นั้นจะมี 2 รูปแบบคือ เวลาที่เข้าถึงข้อมูลหรือเปิดไฟล์ขึ้นมาอ่านเท่านั้นโดยยังไม่มีการแก้ไข เรียกว่า access time และเวลาของไฟล์ที่ถูกแก้ไขล่าสุด เรียกว่า modification time)
Syntax	**touch [*options*] *files***
Example	**# touch newfile**

options :

-a, --time=atime, --time=access, --time=use แก้ไขเวลาที่ทำการเปิดไฟล์หรือเข้าถึงไฟล์ (เวลาดังกล่าวคือเวลาที่ทำการเปิดอ่านไฟล์แต่ยังไม่มีการแก้ไขไฟล์)

-c, --no-create ไม่มีการสร้างไฟล์ใดๆ ทั้งสิ้น

-d *time*, --date *time* เปลี่ยนแปลงเวลาของไฟล์ตามที่กำหนดใน time แทนที่เวลาของไฟล์ในปัจจุบัน

-m, --time=mtime, --time=modify แก้ไขเวลาของไฟล์ที่แก้ไขล่าสุด

-r *file*, --reference *file* ใช้เวลาของไฟล์ *file* แทนเวลาของไฟล์เดิมที่ใช้งานอยู่

-t *time* กำหนดเวลาของไฟล์ที่ต้องการ โดยมีรูปแบบในการกำหนดคือ [[*CC*] *YY*] *MMDDhhmm*[.*ss*] โดยที่

MM - The month of the year [01-12].

DD - The day of the month [01-31].

hh - The hour of the day [00-23].

mm - The minute of the hour [00-59].

CC - The first two digits of the year.

YY - The second two digits of the year.

SS - The second of the minute [00-61].

files คือไฟล์ที่ต้องการปรับแต่งข้อมูลเวลา หรือสร้างไฟล์

ตัวอย่างการใช้งาน :

แสดงเวลาของไฟล์ชื่อว่า newfile.txt ก่อนใช้คำสั่ง touch

```
-rw-r--r--  1 root root       17 Jan 20 10:32 newfile.txt
# touch newfile.txt
-rw-r--r--  1 root root       17 Feb  5 09:12 newfile.txt
```

สร้างไฟล์ใหม่ชื่อว่า newfile.txt เมื่อไฟล์ดังกล่าวไม่มีอยู่ในระบบ คำสั่ง touch จะทำการสร้างไฟล์ใหม่ที่ว่างเปล่าขึ้นมาทันที แต่ถ้ามีอยู่แล้วจะทำการแก้ไขเวลาการ access ไฟล์เป็นเวลาใหม่ทันที(เวลาของ myfile.txt เปลี่ยนจาก 17 Jan เป็น 17 Feb)

```
# touch file1 file2 file3
```

สร้างไฟล์ใหม่ที่ว่างเปล่า 3 ไฟล์ชื่อว่า file1 file2 file3

```
# touch -am file3
```

เปลี่ยนค่าเวลา access time(-a) และเปลี่ยนเวลา modification time(-m) ของ file3 เป็นเวลาปัจจุบัน (สมมุติเดิมค่า access time เป็น 12.00 น เมื่อใช้ –m เวลาจะเปลี่ยนเป็น 18.00 สมมุติว่าเวลาตอนนี้คือ 18.00 น เป็นต้น)

```
# touch -r file4 file5
```

เป็นการเปลี่ยนเวลาในการของไฟล์จากเดิม เป็นเวลาใหม่ โดยใช้เวลาจากไฟล์ file4 แทนที่เวลาของ file5 (สมมุติว่าเดิม file5 มีเวลาเป็น 11.00, file4 เท่ากับ 8.30 เวลาใหม่ของ file5 จะเป็น 8.30)

```
# touch -d '1 May 2005 10:22' file8
```

เปลี่ยนเวลาของ file8 เป็น 1 May 2005 10:22

```
# touch -t 199910150910.55 file1
```

เปลี่ยนเวลาของ file1 เป็น ปี 1999 เดือน 10 วันที่ 15 เวลา 9.10 นาที 55 วินาที

(1999=YY, 10=MM, 15=DD, 09=hh, 10=mm, .55=SS)

```
# touch -t 10150910.55 file1
```

เปลี่ยนเวลาของ file1 เป็น เดือน 10 วันที่ 15 เวลา 9.10 นาที 55 วินาที

```
# touch -t 1010150910.55 file1
```

เปลี่ยนเวลาของ file1 เป็น ปี 2010 เดือน 10 วันที่ 15 เวลา 9.10 นาที 55 วินาที

(10=CC, 10=MM, 15=DD, 09=hh, 10=mm, .55=SS)

ลองทดสอบ :

```
# touch -d '14 May' file9
# touch -d '14:24' file9
```

ดูคำสั่งอื่นประกอบ : *date*

คำสั่ง chown

File Properties : **chown**	
คำสั่ง	**chown** เป็นคำสั่งที่ใช้สำหรับเปลี่ยนเจ้าของไฟล์ข้อมูลหรือไดเรคทรอรี
Syntax	**chown [options] newowner files** **chown [options] --reference=filename files**
Example	**# chown test file.txt** **# chown test:users /home/install**

options:-

-c, --changes แสดงผลการเปลี่ยนสิทธิ์ของไฟล์ข้อมูล

-f, --silent, --quiet ไม่แสดงข้อความผิดพลาด ในกรณีที่ไม่สามารถเปลี่ยนสิทธิ์ได้

-v, --verbose แสดงข้อมูลทั้งหมดที่เป็นผลจากคำสั่งดังกล่าว

--reference=*filename* เปลี่ยนสิทธิ์ของไฟล์ โดยใช้สิทธิ์ที่อยู่กับไฟล์แทนการระบุเจ้าของตรงๆ

-R, --recursive เปลี่ยนสิทธิ์ทุกๆ ไดเรคทรอรีย่อย

--help คำสั่งช่วยเหลือการใช้งาน

--version แสดงเวอร์ชันของ cdmod

ตัวอย่างการใช้งาน :

chown nobody data.txt

เปลี่ยนสิทธิ์ไฟล์ชื่อ data.txt มีเจ้าของใหม่เป็นชื่อ nobody

chown somsri:users /home/doc

เปลี่ยนสิทธิ์ไดเรคทรอรีชื่อ /home/doc มีเจ้าของใหม่เป็นชื่อ somsri อยู่ในกลุ่ม users

chown roger file1 file2

เปลี่ยนสิทธิ์ให้ไฟล์ชื่อ file1 file2 มีเจ้าของใหม่เป็น roger

chown -R hiox test

เปลี่ยนสิทธิ์ให้ไดเรคทรอรีชื่อ test มีเจ้าของใหม่เป็นชื่อ hiox ถ้าไดเรคทรอรี test มีไดเรคทรอรีย่อยภายใน คำสั่งนี้จะมีผลคือได้รับสิทธิ์ในไดเรคทรอรีย่อยๆ ไปด้วย

chown -c hiox calc.txt

changed ownership of `cale.txt' to hiox

เปลี่ยนสิทธิ์ให้ไฟล์ชื่อ cale.txt มีเจ้าของใหม่เป็นชื่อ hiox พร้อมแสงผลของการเปลี่ยน

chown john:family picture.jpg

เปลี่ยนสิทธิ์ให้ไฟล์ชื่อ picture.jpg มีเจ้าของใหม่เป็นชื่อ john เป็นสมาชิกในกลุ่ม family

chown –R john:family pictures

เปลี่ยนสิทธิ์ให้ไดเรคทรอรีและไดเรคทรอรีย่อย ชื่อ pictures มีเจ้าของใหม่เป็นชื่อ john เป็นสมาชิกในกลุ่ม family

chown -R user.group /path/to/data

เปลี่ยนสิทธิ์ให้ไดเรคทรอรีชื่อ /path/to/data มีเจ้าของใหม่เป็นชื่อ user เป็นสมาชิกใน

กลุ่ม group การใช้ option –R จะส่งผลให้ subtree ย่อยๆ ใน data มีการเปลี่ยนสิทธิ์ด้วย

chown username somedir

เปลี่ยนสิทธิ์ให้ไดเรคทรอรีชื่อ somedir มีเจ้าของใหม่เป็นชื่อ username แต่จะไม่ส่งผลกับไดเรคทรอรีย่อยๆ ใน somedir

chown -v username somefile

```
changed ownership of 'somefile' to username
```

เปลี่ยนสิทธิ์ให้ไดเรคทรอรีชื่อ somedir มีเจ้าของใหม่เป็นชื่อ username พร้อมแสดงผล

chown -Rv username somedir

```
changed ownership of 'somedir/' to username
changed ownership of 'somedir/boringfile' to username
changed ownership of 'somedir/somefile' to username
```

เปลี่ยนสิทธิ์ให้ไดเรคทรอรีชื่อ somedir มีเจ้าของใหม่เป็นชื่อ username พร้อมแสดงผลการเปลี่ยนแปลง ในไดเรคทรอรีย่อยทั้งหมดด้วย

chown --reference=oldfile newfile

เปลี่ยนสิทธิ์ไฟล์ชื่อ newfile มีเจ้าของใหม่เป็นชื่อและสมาชิก ที่มีสิทธิ์ในไฟล์ชื่อ oldfile อยู่แล้ว (เหมือนการสำเนาสิทธิ์)

ลองทดสอบ :

```
# chown -R root.root .
# chown -R root.root ..
# chown -R root.root /home/test/*
```

ดูคำสั่งอื่นประกอบ : *chmod, ls*

คำสั่ง chgrp

File Properties : **chgrp**	
คำสั่ง	**chgrp** เป็นคำสั่งที่ใช้สำหรับเปลี่ยนกลุ่มเจ้าของไฟล์ข้อมูลหรือไดเรคทรอรี การเปลี่ยนแปลงและแก้ไขจะมีผลกระทบต่อไฟล์ /etc/group
Syntax	**chgrp [*options*] *newgroup files***
Example	**# chgrp newgroup file**

options:-

-c, --changes พิมพ์ข้อความที่มีการเปลี่ยนแปลง

-f, --silent, --quiet ไม่แสดงข้อความผิดพลาดในกรณีที่ไม่สามารถแก้ไขได้

--help ช่วยเหลือในการใช้งาน

-R, --recursive เปลี่ยนกลุ่มในทุกๆ ไดเรคทรอรีย่อยๆ

ตัวอย่างการใช้งาน :

chgrp users file.txt

สมมุติว่า file.txt มีชื่อเจ้าของเป็นกลุ่ม admin เมื่อประมวลผลคำสั่งข้างต้น file.txt จะ

เปลี่ยนเจ้าของกลุ่มใหม่เป็น users แทน
chgrp admin /home/harry
เปลี่ยนกลุ่มของไดเรคทรอรี harry เป็นกลุ่มใหม่ชื่อว่า admin
chgrp -R admin /home/
สมมุติว่าใน /home มีไดเรคทรอรีย่อยคือ test, Tom, Harry และทุกไดเรคทรอรีเป็นสมาชิกกลุ่ม users อยู่ แต่เมื่อประมวลผลคำสั่งข้างต้นจะทำให้ทุกไดเรคทรอรี home และไดเรคทรอรีย่อยๆ มีสมาชิกใหม่เป็น admin ทันที

ลองทดสอบ :

```
# chgrp -hR staff /u
```

ดูคำสั่งอื่นประกอบ : *chmod, chown, id*

คำสั่ง chmod

File Properties : **chmod**	
คำสั่ง	**chmod** เป็นคำสั่งที่ใช้สำหรับเปลี่ยนเจ้าของไฟล์ข้อมูลหรือไดเรคทรอรี
Syntax	**chmod [OPTION]... MODE[,MODE]... FILE...** **or: chmod [OPTION]... OCTAL-MODE FILE...** **or: chmod [OPTION]... --reference=RFILE FILE...**
Example	**# chmod 755 myfile** **# chmod o+w test.txt**

options:-

-c, --changes พิมพ์ข้อความการเปลี่ยนแปลงที่เกิดจากการใช้คำสั่ง chmod

-f, --silent, --quiet ไม่แสดงข้อความผิดพลาดในกรณีที่ไม่สามารถแก้ไขได้

--help ช่วยเหลือในการใช้งาน

-R, --recursive เปลี่ยนกลุ่มในทุกๆ ไดเรคทรอรีย่อยๆ

คำสั่ง chmod เป็นคำสั่งที่ใช้สำหรับกำหนดสิทธิของผู้ใช้ในกลุ่มต่างๆ ให้สามารถเข้าถึงไฟล์ต่างๆ ได้ ระบบจะแบ่งกลุ่มผู้ใช้ออกเป็น 3 กลุ่มด้วยกัน

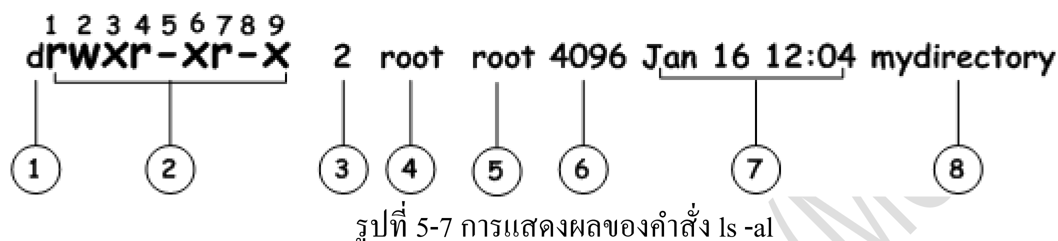
1. เจ้าของ (owner) คือผู้ที่สร้างไฟล์ขึ้นมา แทนด้วยตัวอักษร (u)
2. กลุ่ม (group) คือ กลุ่มผู้ใช้ ผู้ใช้ที่อยู่กลุ่มเดียวกันสามารถใช้งานไฟล์ของเพื่อนร่วมกลุ่มได้ แทนด้วยตัวอักษร (g)
3. ผู้อื่น (other) คือ บุคคลที่ไม่ได้อยู่ในกลุ่ม หรือ ไม่ใช่เจ้าของไฟล์ แทนด้วยตัวอักษร (o)

ผู้ใช้สามารถกำหนดสิทธิต่างๆให้กับไฟล์ได้ โดยแบ่งออกเป็น 3 อย่างด้วยกัน

1. read คือสิทธิในการอ่านไฟล์ แทนด้วยตัวอักษร (r)
2. write คือ สิทธิในการแก้ไขไฟล์ แทนด้วยตัวอักษร (w)
3. execute คือ สิทธิในการใช้งานหรือประมวลผลไฟล์ แทนด้วยตัวอักษร (x)

การเพิ่มและลดสิทธิ์จะใช้เครื่องหมาย + (เพิ่มสิทธิ์) และ - (ลดสิทธิ์) จากตัวอย่างข้างล่างแสดงรายละเอียดภายในไดเรคทรอรีด้วยคำสั่ง ls –al

```
[root@localhost install]# ls -al
total 20
drwxr-xr-x   3 root root 4096 Jan 16 12:05 .
drwxr-x---  26 root root 4096 Jan 16 11:29 ..
drwxr-xr-x   2 root root 4096 Jan 16 12:04 mydirectory
-rw-r--r--    1 root root    117 Jan 16 12:04 myfile.txt
lrwxrwxrwx  1 root root      19 Jan 16 12:05 mylink -> /usr/sbin/showmount
```



รูปที่ 5-7 การแสดงผลของคำสั่ง ls -al

จากคำสั่ง ls ข้างต้น ผลลัพธ์ที่ได้แบ่งออกเป็น 8 คอลัมภ์คือ

คอลัมภ์ 1: แสดงรายละเอียดชนิดของไฟล์ ดังนี้

d ไดเรคทรอรี

- Text ไฟล์ธรรมดา

l ลิงค์ไฟล์

p ไฟล์ประเภท pipe

s ไฟล์ socket

c character device

b block device

คอลัมภ์ 2: แสดงสิทธิ์ในการเข้าถึงไฟล์ ดังนี้

อักษรตัวที่ 1-3 บอกว่าเป็นสิทธิของเจ้าของไฟล์

อักษรตัวที่ 4-6 บอกว่าเป็นสิทธิของผู้ใช้ที่อยู่ในกลุ่มเดียวกัน

อักษรตัวที่ 7-9 บอกว่าเป็นสิทธิของผู้ใช้อื่นๆนอกเหนือจาก 2 กลุ่มแรก

รูปแบบการใช้งานแบบไบนารี

ตัวอย่างเช่นการกำหนดสิทธิ์ แบบมี 3 อักษร

ตัวอักษร r มาจาก Read หมายถึง อ่าน(1 = ทำงาน, 0 = ไม่ทำงาน)

ตัวอักษร w มาจาก Write หมายถึง เขียน(1 = ทำงาน, 0 = ไม่ทำงาน)

ตัวอักษร x มาจาก Execute หมายถึง ประมวลผล(1 = ทำงาน, 0 = ไม่ทำงาน)

u
4 2 1
rwx

รูปที่ 5-8 แสดงค่าประจำบิต(เฉพาะผู้ใช้ u)

--- : ไม่มีสิทธิอะไรเลย (เป็น 0 ทุกบิต เลขที่ใช้คือ 0)

--x : ประมวลผลได้อย่างเดียว (เป็น 1 บิตสุดท้าย ค่าที่ได้คือ 1)

r-- : อ่านได้อย่างเดียว (ค่าที่ใช้คือ 4)

rw- : อ่าน และเขียนได้ (ค่าที่ใช้คือ 6 = 4 + 2)

r-x : อ่าน และประมวลผลได้ (ค่าที่ใช้คือ 5 = 4 + 1)

rwx : อ่าน เขียน และประมวลผลได้ (ค่าที่ใช้คือ 7 = 4 + 2 + 1)

ตัวอย่างเช่นการกำหนดสิทธิ์ แบบมี 9 บิต

3 บิตแรก(1-3) หมายถึง เจ้าของ

3 บิตถัดไป(4-6) หมายถึง สมาชิกในกลุ่ม

3 บิตสุดท้าย(7-9) หมายถึง กลุ่มอื่นๆ

u	g	o
4 2 1	4 2 1	4 2 1
rwx	rwx	rwx

รูปที่ 5-9 แสดงค่าประจำบิตทั้ง 9 บิต

rwx------ : เจ้าของไฟล์เท่านั้นที่มีสิทธิทุกอย่าง (ค่าที่ใช้คือ 700)

rxxrwx--- : เจ้าของไฟล์ และสมาชิกกลุ่มเดียวกันมีสิทธิทุกอย่าง (ค่าที่ใช้คือ 770)

rwxr-xr-x : เจ้าของไฟล์มีสิทธิทุกอย่าง ส่วนกลุ่มและคนอื่นสามารถอ่านและประมวลผลได้ (ค่าที่ใช้คือ 755)

r--r--r-- : เจ้าของไฟล์ สมาชิกในกลุ่ม และกลุ่มอื่นๆ สามารถอ่านได้อย่างเดียว (ค่าที่ใช้คือ 444)

รูปแบบการใช้งานแบบสัญลักษณ์ (u, g, o, a = all ทุกคน)

รูปแบบการใช้คำสั่งเปลี่ยนสิทธิ์คือ chmod [u / g / o] [+ / -] [r / w / x] filename เช่น

chmod u+r filename เพิ่มสิทธิ์ของ filename ให้กับเจ้าของไฟล์อ่านได้

chmod ugo-w filename ลดสิทธิ์ในการเขียนไฟล์ ให้กับทุกๆ คนและทุกกลุ่ม

chmod ug+x filename เพิ่มสิทธิ์ในการประมวลผลให้เจ้าของไฟล์และสมาชิกภายในกลุ่ม

คอลัมภ์ 3: จำนวนลิงค์ จะบอกให้ทราบถึงจำนวนไฟล์ที่ลิงค์เข้ากับไดเรคทอรีหรือไฟล์นี้ ถ้าเป็นไดเรคทอรีว่างจะมีค่าระบุไว้เป็น 2

คอลัมภ์ 4: บอกให้ทราบว่าใครเป็นเจ้าของไฟล์หรือไดเรคทอรี

คอลัมภ์ 5: บอกให้ทราบผู้ใช้อยู่ในกลุ่มใด

คอลัมภ์ 6: บอกให้ทราบถึงขนาดของไฟล์ว่ามีความจุเท่าไหร่

คอลัมภ์ 7: วัน-เวลา จะแสดงวันและเวลาในการแก้ไขไฟล์หรือไดเรคทอรีล่าสุด

คอลัมภ์ 8: ชื่อไฟล์ แสดงชื่อไฟล์ หรือไดเรคทอรี

ตัวอย่างการใช้งาน :

chmod 644 file.htm
กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่านและเขียนได้ สมาชิกภายในกลุ่มและกลุ่มอื่นๆ อ่านได้อย่างเดียว

chmod 755 file.cgi
กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่าน เขียนและประมวลผลได้ สมาชิกภายในกลุ่ม อ่านและประมวลผลได้ กลุ่มอื่นๆ สามารถอ่านและประมวลผลได้

chmod 666 file.txt
กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่าน เขียนได้ สมาชิกภายในกลุ่มอ่าน เขียนได้ และกลุ่มอื่นๆ สามารถอ่าน เขียนได้

chmod 664 myfile
กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่านและเขียนได้ สมาชิกภายในกลุ่มอ่าน เขียนได้ และกลุ่มอื่นๆ อ่านได้อย่างเดียว

chmod u+x file
เพิ่มสิทธิ์ให้เจ้าของไฟล์ประมวลผล file ได้

chmod u=rwx,g=rx,o=x file
กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่าน เขียน และประมวลผลได้ สมาชิกภายในกลุ่ม สามารถอ่านและประมวลผลได้ และกลุ่มอื่นๆ ประมวลผลได้ (751)

chmod a-wx,a+r file
อันดับแรก กำหนดสิทธิ์ทุกๆ คน ไม่มีสิทธิ์ในการเขียน และประมวลผล จากนั้น กำหนดให้ทุกๆ คน อ่านได้ (444)

chmod ug+rw mydir
กำหนดสิทธิ์ให้เจ้าของไฟล์และสมาชิกในกลุ่ม สามารถอ่าน เขียนได้

chmod a-w myfile
ลดสิทธิ์ให้ทุกคนไม่สามารถเขียนไฟล์ได้

chmod -R 644 pictures
กำหนดสิทธิ์ให้เจ้าของไฟล์ อ่านและประมวลผลได้ สมาชิกในกลุ่มและกลุ่มอื่นๆ อ่านได้อย่างเดียว ถ้าไดเรคทรอรี pictures มีไดเรคทรอรีย่อยก็ให้กำหนดสิทธิ์ลงไปทุกๆ

```
ไดเรคทรอรีย่อยทั้งหมดด้วย
# chmod a-w myfile
ลดสิทธิ์ให้ทุกคนไม่สามารถเขียนไฟล์ได้
# chmod a-w myfile
ลดสิทธิ์ให้ทุกคนไม่สามารถเขียนไฟล์ได้
```

ลองทดสอบ :

```
# chmod ug=rx mydir
# chmod u+x,g-w,o=r chfile.pl
# chmod 777 file1.txt
# chmod u=rw,g=rw,o=rw filename
# chmod ugo+rwx directory1
# chmod go-rwx directory1
# chmod u+s /bin/file1
# chmod u-s /bin/file1
# chmod g+s /home/public
# chmod g-s /home/public
# chmod o+t /home/public
# chmod o-t /home/public
```

ดูคำสั่งอื่นประกอบ : *chown, getfacl, ls*

คำสั่ง chattr

File Properties : **chattr**	
คำสั่ง	**chattr** เป็นคำสั่งที่ใช้สำหรับเปลี่ยนคุณลักษณะของไฟล์ข้อมูลขั้นสูง
Syntax	**chattr [*options*] *mode files***
Example	**# chattr +a myfile** **# chmod o+w test.txt**

options:-

-R เปลี่ยนคุณลักษณะทั้งหมดในไดเรคทรอรีย่อย

-V แสดงผลลัพธ์ทั้งหมดที่เปลี่ยนแปลง

-v *version* แสดงเวอร์ชันของ chattr

สัญลักษณ์ที่ใช้เพิ่มหรือลดหรือกำหนดคุณลักษณ์ จะใช้เครื่องหมาย + แทนการเพิ่มคุณลักษณะ, - แทนการลด, = แทนการกำหนดคุณลักษณ์เข้าไปโดยตรง

คุณลักษณะที่สามารถกำหนดได้มีดังนี้

A ไม่แก้ไขเวลาในการเข้าปรับปรุงไฟล์

a เพิ่มสิทธิ์ในการเขียนเท่านั้น

c บีบอัดไฟล์

d No dump

i Immutable (ไม่เปลี่ยนรูปคุณลักษณะของไฟล์ คือ ไม่ยอมให้แก้ไขคุณลักษณะของไฟล์ได้นั่นเอง)

S Synchronous updates

s เมื่อลบไฟล์แล้ว จะไม่สามารถกู้กลับคืนมาได้

u ตรงข้ามกับ s

ถ้าต้องการแสดงคุณสมบัติของไฟล์ที่กำหนดด้วยคำสั่ง chattr ไว้จะต้องใช้คำสั่ง lsattr ในการแสดงผล

ตัวอย่างการใช้งาน :

chattr +a /log/system.log
ป้องกันการลบไฟล์ /log/system.log จาก root หรือโพรเซส หรือผู้ใช้ แต่ยังสามารถเขียนข้อมูลเพิ่มเข้าไปได้

lsattr system.log
-----a------- system.log
แสดงผลไฟล์ที่กำหนดคุณลักษณะด้วยคำสั่ง chattr

chattr +i /file/php.ini
rm -f php.ini
rm: cannot remove `php.ini': Operation not permitted
ป้องกันการลบหรือการแก้ไขไฟล์ php.ini จาก root หรือโพรเซส หรือผู้ใช้อื่นๆ

chattr -i /file/php.ini
ลบการป้องกันการลบหรือการแก้ไขไฟล์ php.ini ออก

chattr +c ch.txt
ls -l ch.txt
-rw-r--r-- 1 root root 163 Jan 18 05:23 ch.txt
กำหนดคุณลักษณะของไฟล์เป็นชนิดบีบอัด (ทดสอบผลด้วยการใช้คำสั่ง lsattr)
--------c---- ch.txt

chattr +d file1
ป้องกันการ dump ไฟล์ของ file1 ในกรณีที่อาจจะกำลังทำการสำรองข้อมูล file1 อยู่

chattr +s file1
ลบไฟล์ file1 แบบปลอดภัย โดยไม่สามารถกู้คืนกลับมาได้อีก

chattr -i /file/php.ini
ลบการป้องกันการลบหรือการแก้ไขไฟล์ php.ini ออก

ลองทดสอบ :

```
# lsattr
# lsattr /log/
# lsattr *
# chattr +i /etc/mail/sendmail.cf
# chattr +i /etc/mail/access
# chattr +S file1
# chattr +u file1
```

ดูคำสั่งอื่นประกอบ : *lsattr*

คำสั่ง lsattr

File Properties : **lsattr**	
คำสั่ง	**lsattr** เป็นคำสั่งที่ใช้สำหรับแสดงคุณลักษณะของไฟล์ข้อมูล
Syntax	**lsattr [*options*] [*files*]**
Example	**# lsattr** **# lsattr myfile**

options:-

-R แสดงคุณสมบัติของไฟล์และไดเรคทรอรีย่อยทั้งหมด

-a แสดงคุณสมบัติของไฟล์และไดเรคทรอรีรวมถึงไฟล์ประเภท . และ .. ด้วย

-d แสดงคุณสมบัติเฉพาะไดเรคทรอรีที่กำหนด

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

```
# lsattr
------------- ./myfile.txt
------------- ./a.txt
------------- ./test
------------- ./log.txt
```
แสดงคุณสมบัติของไฟล์ข้อมูลทั้งหมดในไดเรคทรอรีในปัจจุบัน

```
# lsattr myfile
----i-------- myfile
```
แสดงคุณสมบัติของไฟล์ข้อมูลชื่อ myfile

ลองทดสอบ :

```
# lsattr -d /path/to/file
lsattr -d /var/spool/squid
chattr +d /var/spool/squid
```

ดูคำสั่งอื่นประกอบ : *ls, wc*

5.3.5 กลุ่มคำสั่งเกี่ยวกับ File Location

คำสั่ง find

File Location : **find**	
คำสั่ง	**find** เป็นคำสั่งที่ใช้สำหรับค้นหาไฟล์ข้อมูลและไดเรคทรอรี
Syntax	**find [*pathnames*] [*conditions*]**
Example	**# find / -name "filesearch" -print** **# find /home/john -name 'index*'**

conditions and actions :

-amin +n| -n| n ค้นหาไฟล์ที่ถูกเปิดล่าสุดเกิน n นาที(+n), หรือน้อยกว่า n นาที(-n), หรือใช้งานมาแล้วเท่ากับ n นาทีพอดี

-anewer file ค้นหาไฟล์ที่ถูกใช้งานหลังจาก file ถูกแก้ไข

-atime +n| -n| n ค้นหาไฟล์ที่ถูกเปิดล่าสุดเกิน n วัน(+n), หรือน้อยกว่า n วัน(-n), หรือใช้งานมาแล้วเท่ากับ n วันพอดี

-cmin +n| -n| n ค้นหาไฟล์ที่มีการเปลี่ยนแปลงเกิน n นาที(+n), หรือน้อยกว่า n นาที(-n), หรือเปลียนแปลงมาแล้วเท่ากับ n นาทีพอดี

-ctime +n| -n| n ค้นหาไฟล์ที่มีการเปลี่ยนแปลงเกิน n วัน(+n), หรือน้อยกว่า n วัน(-n), หรือเปลียนแปลงมาแล้วเท่ากับ n วันพอดี

-exec command{ } \ ; เป็นออพชัน ที่ใช้ต่อหลังคำสั่ง find เพื่อให้สามารถใช้คำสั่ง find ได้ยืดหยุ่นขึ้น คำสั่งนี้จะทำงานหลังจากที่คำสั่ง find ทำงานแล้ว เช่น find / -name "test*" –exec ls –l {} \; ไฟล์ที่ขึ้นต้นด้วย test จะถูกแสดงผลด้วย ls –l (command = ls -l)

-follow แสดง symbolic links และ track ของไดเรคทรอรี

-fstype type ค้นหาโดยระบุชนิดของไฟล์ เช่น minix, ext, ext2, xia, msdos, umsdos, vfat, proc, nfs, iso9660, hpfs, sysv, smb, and ncpfs

-gid ค้นหาไฟล์โดยกำหนด ID กลุ่มผู้ใช้ (group ID)

-group gname ค้นหาไฟล์โดยกำหนดชื่อกลุ่มผู้ใช้ (group name)

-name pattern ใส่คำค้นในการค้นหาแบบ case-sensitive คำค้นควรจะอยู่ภายใต้เครื่องหมาย ' ' หรือ " "

-iname pattern เหมือนกับ –name แต่ค้นหาแบบ case-insensitive

-lname pattern ค้นหาไฟล์ที่เป็น symbolic links แบบ case-sensitive

-ilname pattern เหมือน –lname หาแบบ case-insensitive

-path pattern ค้นหาไฟล์โดยการระบุเส้นทางเริ่มต้นในการค้นหา

-regex pattern ใช้ regular expression ช่วยในการค้นหาแบบ case-insensitive

-iregex pattern เหมือน regex หาแบบ case-insensitive

-links n ค้นหาไฟล์ที่มีจำนวนลิงค์เท่ากับ n

-maxdepth num ค้นหาเข้าไปในไดเรคทรอรีลึกไม่เกินค่า num

-mindepth num ค้นหาเข้าไปในไดเรคทรอรีลึกไม่ต่ำกว่าค่า num

-mmin +n| -n| n ค้นหาไฟล์ที่แก้ไขล่าสุดเกิน n นาที(+n), หรือน้อยกว่า n นาที(-n), หรือใช้งานมาแล้วเท่ากับ n นาทีพอดี

-mtime +n| -n| n ค้นหาไฟล์ที่แก้ไขล่าสุดเกิน n วัน(+n), หรือน้อยกว่า n วัน(-n), หรือใช้งานมาแล้วเท่ากับ n วันพอดี

-ok command { }\; ทำงานคล้ายคำสั่ง –exec แต่จะมี prompt ให้ตอบโต้กับโปรแกรมได้ เมื่อกด yes โ ปรแกรมจึงจะทำงานต่อไป

-perm nnn ค้าหาไฟล์ที่มี permission ตามค่าที่กำหนด คือ nnn ตัวอย่างเช่น –perm 644 ซึ่งตรงกับ -rw-rw-r—

-print เมื่อค้นหาพบกับไฟล์ที่ต้องการจะแสดงผลออกโดยการพิมพ์ออกจอภาพ แบบ full pathnames

-size n ค้นหาไฟล์ที่มีขนาด n blocks

-type c ค้นหาไฟล์ตามชนิดที่ระบุใน c ค่า c สามารถเป็นไปได้หลายแบบคือ b (block special file), c (character special file), d (directory), p (fifo or named pipe), l (symbolic link), s (socket), or f (plain file)

ตัวอย่างการใช้งาน :

find / -name game
ค้นหาไฟล์หรือไดเรคทรอรีชื่อว่า game ทุกไดเรคทรอรีในระบบ

find $HOME -print
แสดงรายชื่อไฟล์ทุกไฟล์ในไดเรคทรอรี home

find / -name hello.pl
ค้นหาไฟล์ชื่อว่า hello.pl โดยเริ่มค้นหาตั้งแต่ไดเรคทรอรี / (หรือค้นหาทุกๆ ไดเรคทรอรีที่มีทั้งหมดในระบบ)

find / -name hello*
ค้นหาไฟล์ที่ขึ้นต้นด้วยคำว่า hello แล้วตามด้วยอักษรอะไรก็ได้ เช่น hello1, helloworld, hello_world, hello-t, hello*, hello$

find /bin -size 626100c
ค้นหาไฟล์ที่มีขนาดเท่ากับ 626100 ในไดเรคทรอรี /bin

find /usr/local/ -name '*Doc*'
ค้นหาไฟล์ที่มีชื่อขึ้นต้นและลงท้ายด้วยอักษรอะไรก็ได้ กี่ตัวก็ได้ แต่ตรงกลางต้องมีอักษรว่า Doc เท่านั้น เริ่มต้นค้นหาไฟล์ในไดเรคทรอรี /usr/local

find /usr/local/ -name '*Doc*' -exec rm {} \;
ค้นหาไฟล์ที่มีชื่อขึ้นต้นและลงท้ายด้วยอักษรอะไรก็ได้ กี่ตัวก็ได้ แต่ตรงกลางต้องมีอักษรว่า Doc เท่านั้น เริ่มค้นหาไฟล์ในไดเรคทรอรี /usr/local เมื่อค้นหาเจอแล้วให้ทำคำสั่งลบไฟล์ที่หาเจอนั้นทิ้งให้หมด

find /usr/local/ -mtime +5
ค้นหาไฟล์ที่ถูกแก้ไขเกินกว่า 5 วัน

find /usr/local/ -mtime -5
ค้นหาไฟล์ที่ถูกแก้ไขต่ำกว่า 5 วัน

find /usr/local/ -mtime +365 -exec rm {} \;

ค้นหาไฟล์เก่ากว่า 1 ปีลงไป และลบไฟล์เหล่านั้นทิ้ง

find /home -user joe

ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรี /home เลือกเฉพาะไฟล์ของผู้ใช้ชื่อ joe เท่านั้น

find /var/spool -mtime +60

ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรี /var/spool ที่ถูกแก้ไขมาแล้วไม่ต่ำกว่า 60 วัน

find $HOME -mtime 0

ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรี /home โดยไฟล์นั้นถูกแก้ไขมาแล้วอย่างน้อย 24 ชั่วโมง

find . -perm -664

ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรีปัจจุบัน ซึ่งไฟล์ที่ค้นหามีเงื่อนไขคือ เจ้าของอ่านและเขียนได้ สมาชิกในกลุ่มสามารถอ่านเขียนได้ กลุ่มอื่นๆอ่านได้อย่างเดียว

find . -perm -220 หรือ **find . -perm -g+w,u+w**

ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรีปัจจุบัน ซึ่งไฟล์ที่ค้นหามีเงื่อนไขคือ เจ้าเขียนได้ สมาชิกในกลุ่มสามารถเขียนได้ กลุ่มอื่นๆ ไม่สามารถทำอะไรได้

find /home/john -name 'index*'

ค้นหาไฟล์ชื่อว่า index และตามด้วยอะไรก็ได้ ภายใต้ไดเรคทรอรี /var/john

find /home/tophy -iname 'index*'

ค้นหาไฟล์ชื่อว่า index และตามด้วยอะไรก็ได้ โดยไม่สนใจตัวอักษรว่าเล็กหรือใหญ่ ภายใต้ไดเรคทรอรี /var/john ผลลัพธ์ที่ได้จะประกอบไปด้วยไฟล์ที่ขึ้นต้นด้วย Index ตัวอักษรถัดไปจะเป็นอะไรก็ได้ และมีจำนวนกี่อักษรก็ได้

find / -name '*.mp3' -size -5000k

ค้นหาไฟล์เริ่มจากตำแหน่ง / โดยค้นหาไฟล์ที่ลงท้ายด้วย .mp3 และมีการกำหนด parameter -size เข้าไปเพื่อระบุว่าค้นหาไฟล์ที่มีขนาดน้อยกว่า 5MB (-5000k)

find / -size +10000k

ค้นหาไฟล์ใดๆ ก็ได้ที่มีขนาดไฟล์มากกว่า 10MB (+10000k)

find /home/tony -amin -10 -name '*.java'

ค้นหาไฟล์จากตำแหน่ง /home/tony ชื่อไฟล์ที่ลงท้ายด้วย .java ที่ access ไปเมื่อ 10 นาทีที่แล้ว

find /home/tony -atime -2 -name '*.java'

ค้นหาไฟล์จาก /home/tony ไฟล์ที่ลงท้ายด้วย .java ที่ access ไม่เกิน 2 วัน

find /home/tony -mmin -10 -name '*.java'

ค้นหาไฟล์จาก /home/tony ไฟล์ที่ลงท้ายด้วย .java ที่แก้ไขมาแล้วไม่เกิน 10 นาที

find /home/tony -mtime -2 -name '*.java'

ค้นหาไฟล์จาก /home/tony ไฟล์ที่ลงท้ายด้วย .java ที่แก้ไขมาแล้วไม่เกิน 2 วัน

find / - name 'king*' -exec ls -l {} \;

ค้นหาไฟล์ที่มีชื่อขึ้นต้นว่า king เมื่อค้นหาเสร็จให้แสดงผลด้วย ls –l

find /work -name 'memo*' -user ann -print

ค้นหาไฟล์ที่ขึ้นต้นด้วย memo โดยมีผู้ใช้ชื่อ ann เป็นเจ้าของไฟล์

```
# find / -type d -name 'man*' -print
```

ค้นหาไดเรคทรอรีที่ขึ้นต้นด้วย man

```
# find . \! -name '[A-Z] *' -exec lpr { } \;
```

ค้นหาไฟล์ที่ไม่ได้ขึ้นต้นด้วยอักษร A-Z ที่เป็นตัวใหญ่ เมื่อพบพิมพ์ออกเครื่องพิมพ์ (เครื่องหมาย ! แสดงว่าค่าที่ได้จะเป็นตรงกันข้าม)

```
# gzip `find . \! -name '*.gz' -print`
```

ค้นหาไฟล์ที่ไม่มีส่วนขยายเป็น .gz เมื่อพบให้ทำการบีบอัดข้อมูลทันที

```
# find / -size 0 -ok rm { } \;
```

ค้นหาไฟล์ที่ไม่มีข้อมูล(ไฟล์ว่าง) เมื่อพบให้ทำการลบออกจากระบบ โดยมีการยืนยันด้วยว่าจะให้ทำการลบหรือไม่

```
# find . -path './kt[0-9]'
```

ต้องการพิมพ์ไฟล์ชื่อที่ขึ้นต้นด้วย kt* ในไดเรคทรอรีปัจจุบัน

```
# find . -regex './kt[0-9]'
```

ต้องการพิมพ์ไฟล์ชื่อที่ขึ้นต้นด้วย kt* ในไดเรคทรอรีปัจจุบัน

```
# find . -size +500000 -print
```

ค้นหาไฟล์ใดๆ ก็ได้ที่มีขนาดไฟล์ใหญ่กว่า 500MB

ลองทดสอบ :

```
# find /home/john -name 'index*' 2>/dev/null
# find /book -print | xargs grep '[Nn] utshell'
# find . -name 'kt[0-9]'
# find / -name Chapter1 -type f -print
# find . -type d -name build
# find . -type f -name "*.java" -exec grep -l StringBuffer {} \;
# find . -type f -name "*.java" -exec grep -il string {} \;
# find /usr/local -name "*.html" -type f -exec chmod 644 {} \;
# find htdocs cgi-bin -name "*.cgi" -type f -exec chmod 755 {} \;
# find . -iname foo -type d
# find . -iname foo -type f
# find . -maxdepth 1 -name *.jpg -print -exec convert
# find / -perm -u+s
```

ดูคำสั่งอื่นประกอบ : *chmod, cpio, locate, ls, sh, whereis, which*

คำสั่ง which

File Location : **which**	
คำสั่ง	**which** เป็นคำสั่งที่ใช้สำหรับค้นหาตำแหน่งที่อยู่ของคำสั่งหรือไฟล์ข้อมูลที่ลงทะเบียนไว้
Syntax	which [*options*] [--] [*commands*]
Example	# which find

options :

-v, -V, --version แสดงเวอร์ชันของคำสั่ง

--help ช่วยเหลือการใช้คำสั่ง

ตัวอย่างการใช้งาน :

```
# which ls
alias ls='ls --color=tty'
        /bin/ls
```
ค้นหาตำแหน่งของคำสั่ง ls (ในที่นี้อยู่ที่ /bin/)

```
# which perl
/usr/bin/perl
```
ค้นหาตำแหน่งของคำสั่ง perl (ในที่นี้อยู่ที่ /usr/bin/)

ลองทดสอบ :

```
# which sshd.conf
# which init
```

ดูคำสั่งอื่นประกอบ : *find, whereis*

คำสั่ง whereis

File Location : **whereis**	
คำสั่ง	**whereis** เป็นคำสั่งที่ใช้สำหรับค้นหาตำแหน่งที่อยู่ของคำสั่งหรือไฟล์ข้อมูลที่ลงทะเบียนไว้ ข้อมูลที่ค้นได้จะประกอบด้วย ที่อยู่ของคำสั่ง, ไฟล์ต้นฉบับ, คู่มือ
Syntax	**which [*options*] *filename***
Example	**# which find**

options :

-b ค้าหาเฉพาะไฟล์หรือคำสั่งที่สามารถประมวลผลได้เท่านั้น (executable file)

-m ค้าหาเฉพาะไฟล์คู่มือเท่านั้น

-s ค้าหาเฉพาะโปรแกรมต้นฉบับเท่านั้น

-u ค้าหาเฉพาะไฟล์ที่ผิดปกติ เช่น ไฟล์ดังกล่าว สามารถประมวลผลได้ และมีโปรแกรมต้นฉบับ แต่ไม่มีคู่มือให้ คำสั่งดังกล่าวจะใช้คู่กับออพชัน –m, b, s อย่างใดอย่างหนึ่ง เช่น whereis –m –u * คำสั่งดังกล่าวจะค้นหาไฟล์ทุกไฟล์ในไดเรคทรอรีปัจจุบัน แสดงเฉพาะไฟล์ที่ไม่มีคู่มือเท่านั้น

-B directory กำหนดขอบเขตการค้นหาเฉพาะไฟล์หรือคำสั่งที่สามารถประมวลผลได้เท่านั้น

-M directory กำหนดขอบเขตการค้นหาเฉพาะไฟล์หรือคำสั่งที่เป็นคู่มือเท่านั้น

-S directory กำหนดขอบเขตการค้นหาเฉพาะไฟล์หรือคำสั่งที่เป็นโปรแกรมต้นฉบับเท่านั้น

-f แสดงผลลัพธ์ที่ค้นหาได้ จะต้องใช้ร่วมกับออพชัน –M, -S, B เสมอ

--help ช่วยเหลือการใช้คำสั่ง



หมายเหตุ: คำสั่ง whereis ค้นได้เฉพาะคำสั่งหรือไฟล์ที่ระบุไว้ใน PATH (path คือตำแหน่งเส้นทางที่บอกให้ระบบปฏิบัติการทราบว่าจะประมวลผลคำสั่งในตำแหน่งใดในระบบ) เท่านั้น หากต้องการค้นหาคำสั่งหรือไฟล์ที่ไม่ได้ลงทะเบียนไว้ ต้องใช้คำสั่ง find แทน

ตัวอย่างการใช้งาน :

whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz
ค้นหาตำแหน่งของคำสั่ง ls จากตัวอย่าง คำสั่งดังกล่าวจะแสดงที่อยู่ของคำสั่ง ls พร้อมกับคู่มือ แต่ไม่มีโปรแกรมต้นฉบับ

whereis -b ls
ls: /bin/ls
ค้นหาตำแหน่งของคำสั่ง ls โดยแสดงผลเฉพาะไฟล์ที่ประมวลผลได้เท่านั้น

whereis -m ls
ls: /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz
ค้นหาตำแหน่งของคำสั่ง ls โดยแสดงผลเฉพาะไฟล์คู่มือเท่านั้น

whereis -u -M /usr/man/man1 -f *
alsacard: /bin/alsacard
alsaunmute: /bin/alsaunmute
arch: /bin/arch
awk: /bin/awk /usr/bin/awk /usr/libexec/awk /usr/share/awk
ค้นหาเฉพาะคู่มือการใช้งานคำสั่งทั้งหมดที่อยู่ในไดเรคทรอรีปัจจุบัน

cd /bin; whereis -u -M /usr/man/man1 -S /usr/src -f *
tcsh: /bin/tcsh
touch: /bin/touch
tracepath: /bin/tracepath /usr/sbin/tracepath
tracepath6: /bin/tracepath6 /usr/sbin/tracepath6
traceroute: /bin/traceroute
traceroute6: /bin/traceroute6
tracert: /bin/tracert
true: /bin/true
...
ค้นหาคู่มือและโปรแกรมต้นฉบับ ทั้งหมดที่อยู่ในไดเรคทรอรี /bin

ลองทดสอบ :

whereis -u -M /usr/man/man1 -S /usr/src -B /bin -f *

ดูคำสั่งอื่นประกอบ : *find, which*

คำสั่ง locate

File Location : **locate**	
คำสั่ง	**locate** เป็นคำสั่งที่ใช้สำหรับหาที่อยู่หรือฐานข้อมูลที่ต้องการในระบบ
Syntax	**locate [*options*] *pattern***
Example	**# locate perl**

options :

-d *path*, --database=*path* ค้นหาข้อมูลโดยการระบุตำแหน่งที่ตั้งของฐานข้อมูล ถ้าต้องการค้นหามากกว่า 1 คำใช้ , คั่น เช่น http, ssh

-i pattern เป็นการค้นหาแบบไม่สนใจตัวอักษรเล็กหรือใหญ่

-q, --quiet ไม่ต้องแสดงข้อผิดพลาดจากการค้นหา เช่น permission ไม่ให้อ่าน

-l, --limit, -n LIMIT กำหนดจำนวนในการค้นหา

-r, --regexp REGEXP กำหนดการค้นหาด้วยการใช้ regular expression

-h, --help คำสั่งช่วยเหลือในการใช้คำสั่ง

--version แสดงเวอร์ชันของ locate

pattern คือข้อความที่ต้องการค้นหา

ตัวอย่างการใช้งาน :

```
# locate perl
/usr/bin/find2perl
/usr/bin/foomatic-perl-data
/usr/bin/perl
/usr/bin/perl5.8.8
/usr/bin/perlbug
/usr/lib/perl5/5.8.5/i386-linux-thread-multi
/usr/lib/perl5/5.8.5/i386-linux-thread-multi/CORE
```
หาตำแหน่งที่ติดตั้งของโปรแกรม perl ที่ติดตั้งในระบบ

locate tomcat.sh

ค้นหาตำแหน่งของไฟล์ tomcat.sh ที่ติดตั้งในระบบ

locate -i springframework

ค้นหาตำแหน่งของไดเรคทรอรีชื่อ springframework

locate "*.dat" -q

ค้นหาตำแหน่งไฟล์ชื่ออะไรก็ได้แต่ต้องมีส่วนขยายเป็น .dat ถ้ามีข้อผิดพลาดจากการค้นหาเช่น อาจจะไม่สามารถอ่านไฟล์ที่มี permission ไม่ให้อ่านได้ จะไม่แสดงความผิดพลาดนั้นออกมา

locate index.html -l 1

ค้นหาไฟล์ index.html การค้นหาจะค้นหาเฉพาะคำนี้เท่านั้น

$locate "MySql*"

ค้นหาตำแหน่งที่ติดตั้งของไฟล์หรือไดเรคทรอรีที่ขึ้นต้นด้วย MySql อักษรที่เหลือเป็น

อะไรก็ได้ และมีกี่ตัวอักษรก็ได้
locate -r "/Movie.*\.avi"
ค้นหาไฟล์ที่ชื่อว่า movie ส่วนขยาย .avi โดยใช้ regular expression

ลองทดสอบ :

```
# locate INDEX.HTML -i
# locate "*.c" -n 10
# locate -d ~/.DB_locatedb sql_db
# locate index.html -l 1
```

ดูคำสั่งอื่นประกอบ : *find, whereis, xargs*

5.3.6 กลุ่มคำสั่งเกี่ยวกับ File Text Manipulation

คำสั่ง grep

File Text Manipulation : **grep**	
คำสั่ง	**grep** เป็นคำสั่งที่ใช้สำหรับค้นหาข้อความที่ต้องการในไฟล์ข้อมูล
Syntax	***grep [options] pattern [files]***
Example	**# grep "unix" *.htm** **# grep root /etc/passwd**

options:-

-c นับจำนวนคำที่ต้องการค้นหา

-i, --ignore-case ค้นหาโดยไม่สนใจตัวอักษรว่าตัวเล็กหรือตัวใหญ่

-l, --files-with-matches แสดงรายชื่อของไฟล์ที่มีข้อมูลตรงกับคำที่ต้องการค้นหา

-n แสดงหมายเลขบรรทัด

-v ไม่แสดงบรรทัดที่มีคำที่ต้องการ

-E ระบุรูปแบบในการค้นหา โดยสามารถระบุรูปแบบ (pattern) ในการค้นหาได้หลายแบบ แบบที่นิยมกันอย่างหนึ่งคือใช้ REGULAR EXPRESSION ในที่นี้ต้องใส่ option -E โดยสามารถใช้ REGULAR EXPRESSION เช่นเดียวกับเวลาที่ใช้ในการเขียนโปรแกรมภาษาต่างๆ

-r ค้นหาข้อความที่ต้องการในไดเรคทรอรีย่อยทั้งหมด

-w, --word-regexp ค้นหาเฉพาะคำที่กำหนดใน regexp เท่านั้น

--color สั่งแสดงสีของข้อความที่ต้องการค้นหา

ตัวอย่างการใช้งาน :

ตัวอย่างไฟล์ /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```

grep root /etc/passwd

```
operator:x:11:0:operator:/root:/sbin/nologin
abc:x:999:501::/root/abc:/bin/bash
```

ต้องการหาคำว่า root ในไฟล์ /etc/passwd ว่ามีอยู่ในบรรทัดใดบ้าง ผลลัพธ์ที่ได้จะแสดงเฉพาะบรรทัดที่มีคำว่า root

grep -n root /etc/passwd

```
1:root:x:0:0:root:/root:/bin/bash
12:operator:x:11:0:operator:/root:/sbin/nologin
37:abc:x:999:501::/root/abc:/bin/bash
```

ค้นหาคำว่า root ผลลัพธ์ที่ได้จะมีเลขที่บรรทัดกำกับมาด้วย ในที่นี้ คือ 1, 12 และ 37 ก็คือบรรทัดที่ 1, 12, 37 นั่นเอง

grep -v root /etc/passwd

```
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

ผลลัพธ์ที่ได้จะแสดงบรรทัดอื่นๆ ที่ไม่มีคำว่า root

grep -E ^ssh /etc/passwd

```
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
```

จากตัวอย่างเราใช้เครื่องหมาย ^ เพื่อระบุว่าตัวอักษรที่ตามมานั้นเป็นตัวอักษรที่ต้องการค้นหาในไฟล์

grep root /etc/*

```
/etc/aliases:postmaster:     root
/etc/aliases:bin:            root
/etc/aliases:daemon:         root
/etc/aliases:adm:            root
```

เมื่อต้องการค้นหาข้อความที่อยู่ในหลายๆ ไฟล์ สามารถทำได้โดยระบุชื่อไฟล์เป็น * ผลลัพธ์ที่ได้จะแสดงชื่อไฟล์นำหน้าบรรทัดที่มีคำว่า root

grep 'Ti[nm]a Jones' staff-listing

ต้องการหาคำว่า Tina Jones หรือ Tima Jones ในไฟล์ staff-listing

grep -v bash /etc/passwd | grep -v nologin

```
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
news:x:9:13:news:/etc/news:
```

ต้องการหาบรรทัดที่ไม่มีข้อความว่า bash ปรากฏอยู่ในไฟล์ /etc/passwd จากนั้นทำการส่งข้อมูลที่ได้ให้กับ grep ผ่านไปป์(อ่านโดยละเอียดเพิ่มเติมในหัวข้อ คำสั่ง ไปป์) อีกครั้งโดย ครั้งหลังนี้ไม่ต้องการบรรทัดที่มีข้อความว่า nologin อยู่ด้วย(ผลลัพธ์ที่ได้

จะต้องเป็นข้อความที่ไม่มีทั้ง bash และ nologin ปรากฏ)

```
# grep -c root /etc/passwd
3
```

ต้องการนับจำนวนคำว่า root ที่พบในไฟล์ passwd ผลที่ได้คือ 3

```
# grep ^root /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

ต้องการหาคำที่ขึ้นต้นบรรทัดว่า root ที่อยู่ในไฟล์ passwd

```
# grep [yf] /etc/group
```

ต้องการหาข้อความที่มีตัวอักษร y หรือ f ตัวใดตัวหนึ่งปรากฏอยู่ในไฟล์ /etc/group

```
# grep '\<c...h\>' /usr/share/dict/words
cauch
cheth
chich
cinch
cirrh-
clach
```

ต้องการหาคำที่ขึ้นต้นด้วย c และลงท้ายด้วย h สามารถใช้ … แทนค่าอะไรก็ได้ตรงกลางแต่ความยาวทั้งหมดรวมแล้วต้องไม่เกิน 6 ตัวอักษร คือ c-f

```
# grep '\<c.*h\>' /usr/share/dict/words
conourish
contra-approach
contradistinguish
controllable-pitch
conveth
....
```

ต้องการหาคำที่ขึ้นต้นด้วย c และลงท้ายด้วย h แต่ความยาวของข้อความไม่จำกัด

```
# grep -i "ftp" /etc/passwd
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

ต้องการหาคำว่า ftp โดยไม่สนใจขนาดตัวอักษรว่าจะเป็นตัวเล็กหรือตัวใหญ่(Ftp, FTP, fTP, ftp สามารถค้นหาได้ทั้งหมด) ที่อยู่ในไฟล์ passwd

```
# grep -r "192.168.1.5" /etc/
```

ต้องการค้นหาคำที่มีหมายเลขไอพี 192.168.1.5 ในไดเรคทรอรี /etc/ รวมถึงไดเรคทรอรีย่อยทั้งหมดที่อยู่ใน /etc ด้วย

```
# grep -w root /etc/group
```

ต้องการหาคำเฉพาะบรรทัดที่มีคำว่า root เท่านั้น(root1, rootroot จะไม่ทำการค้นหาให้)

```
# egrep -w 'root|ftp' /etc/passwd
```

ต้องการค้นหาคำมากกว่า 1 คำที่มีความแตกต่างกัน ให้ใช้ egrep แทน

```
# dmesg | egrep '(s|h)d[a-z]'
  ide1: BM-DMA at 0x1058-0x105f, BIOS settings: hdc:DMA, hdd:pio
hdc: VMware Virtual IDE CDROM Drive, ATAPI CD/DVD-ROM drive
SCSI device sda: 16777216 512-byte hdwr sectors (8590 MB)
sda: Write Protect is off
```

```
sda: Mode Sense: 5d 00 00 00
sda: cache data unavailable
sda: assuming drive cache: write through
SCSI device sda: 16777216 512-byte hdwr sectors (8590 MB)
```

เป็นคำสั่งที่ใช้แสดงข้อมูลของ disk drive (sda=SCSI, had=IDE) คำสั่ง dmesg จะแสดงรายละเอียดเกี่ยวกับระบบทั้งหมดออกมา ซึ่งมีจำนวนมาก ข้อมูลที่ได้จะใช้ egrep ทำการค้นหาอักษรตัวแรกที่ขึ้นต้นด้วย s หรือ h ตามด้วย d และลงท้ายเป็นอะไรก็ได้ จาก a-z ทุกข้อความที่ค้นพบจะมีคำที่ขึ้นต้นด้วย sd หรือ hd เสมอ

cat /proc/cpuinfo | grep -i 'Model' หรือ **grep -i 'Model' /proc/cpuinfo**

```
model           : 14
model name      : Genuine Intel(R) CPU          T2130  @ 1.86GHz
```

ผลลัพธ์จากคำสั่ง cat ที่แสดงข้อมูลในไฟล์ cpuinfo จะถูกส่งต่อมายัง grep จากนั้น grep จะทำการค้นหาคำว่า Model โดยไม่สนใจตัวอักษรว่าจะมีขนาดเล็กหรือใหญ่

grep -l 'main' *.c

แสดงรายชื่อของไฟล์ที่มีคำว่า main อยู่ในไฟล์ทุกไฟล์ที่มีส่วนขยายเป็น .c ในไดเรคทอรีปัจจุบัน

grep --color root /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
abc:x:999:501::/root/abc:/bin/bash
```

แสดงสีของข้อความที่ต้องการค้นหาในที่นี้คือ root

ps -ef|grep ssh

```
root      4637     1  0 08:38 ?        00:00:00 /usr/sbin/sshd
root      5041  4637  0 08:48 ?        00:00:02 sshd: root@pts/1
root      6110  5043  0 10:06 pts/1    00:00:00 grep ssh
```

คำสั่ง ps –ef จะแสดงการทำงานของโพรเซสทั้งหมดในระบบ grep จะทำการค้นหาเฉพาะโพรเซส ssh มาแสดงผลเท่านั้น

ลองทดสอบ :

```
# grep root < /etc/passwd
# grep 'tha*' CREDITS
# grep 'sm[ai]' CREDITS
# grep '[a-z]' CREDITS
# grep 'Sm[^i]' CREDITS
# grep 'Linus$' CREDITS
# grep '\[' CREDITS
# grep [0-9] /var/log/messages
# grep Aug -R /var/log/*
```

ดูคำสั่งอื่นประกอบ : *egrep, fgrep, sed, sh*

คำสั่ง cut

File Text Manipulation : **cut**	
คำสั่ง	**cut** เป็นคำสั่งที่ใช้สำหรับคัดแยกคำของแต่ละบรรทัดในไฟล์ข้อมูล
Syntax	**cut *options [files]***
Example	**# cut -d: -f1,5 /etc/passwd**

options:-

-b, --bytes=LIST คัดแยกตัวอักษรในตำแหน่งที่ระบุหลังออพชัน

-c, --characters=LIST คัดแยกตัวอักษรคล้ายออพชัน –b (คอลัมภ์แรกเริ่มที่ 1)

-d, --delimiter=DELIM คัดแยกตัวอักษรหรือคำ โดยใช้สัญลักษณ์ระบุการแบ่งแยก (delimiter) ตัวอักษรหรือคำ จำเป็นต้องใช้ควบคู่กับออพชัน –f เสมอ (สัญลักษณ์ที่ถูกใช้ทั่วไปคือตัวอักษรว่าง แต่ถ้าต้องการใช้ตัวอักษรพิเศษ เช่น \, ‘ ต้องอยู่ภายใต้เครื่องหมาย ‘ ‘)

-f, --fields=LIST คัดแยกตัวอักษรที่ระบไว้ใน LIST

-s, --only-delimited ไม่ต้องแสดงข้อความในบรรทัดที่ไม่มีสัญลักษณ์ delimiter อยู่

--output-delimiter=string ใช้ข้อความใน string เป็นสัญลักษณ์ delimter

--complement แสดงข้อมูลที่ไม่ได้ถูกเลือก

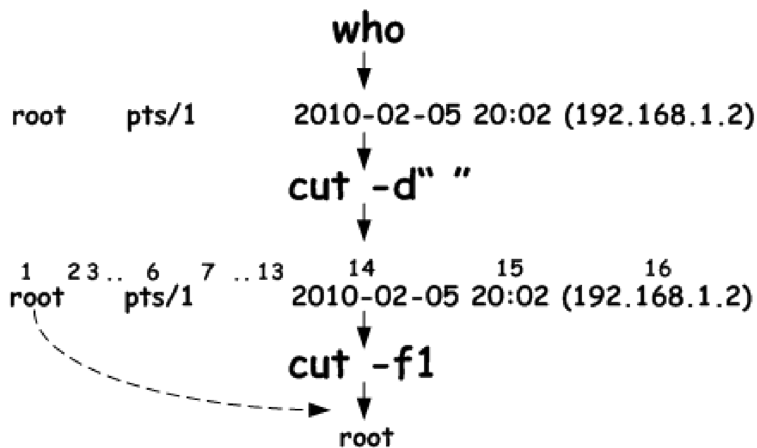
--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

ทดสอบคำสั่ง who

```
root     pts/1       2010-02-05 20:02 (192.168.1.2)
# who | cut -d" " -f1
root
```

จากตัวอย่าง คำสั่ง who จะแสดงรายชื่อผู้ที่ใช้งานระบบอยู่ ในที่นี้คือ root คำสั่งดังกล่าวจะแสดงข้อมูลชื่อผู้ใช้(root), เทอร์มินัลที่ใช้ (pts/1), วันและเวลาที่ทำการล็อคอินเข้าใช้งาน (2010-02-05 20:02) และหมายเลขไอพี ข้อมูลที่ได้จาก who จะถูกส่งไปให้กับ cut ผ่านไปป์ (ดูคำสั่ง ไปป์ เพิ่มเติม) คำสั่ง cut จะทำการตัดคำโดยแต่ละคำถูกคัดแยกด้วยช่องว่าง (ระบุไว้ใน –d" ") สุดท้ายจะนำเอาเฉพาะข้อมูลในคอลัมถ์ที่ 1 มาแสดงผลเท่านั้น(ระบุไว้ใน –f1) ดังรูปที่ 5-10



รูปที่ 5-10 แสดงการใช้คำสั่ง cut –d" " –f1



ข้อควรระวัง: การใช้ตัว delimeter นั้นอาจจะทำให้เข้าใจผิดได้ง่าย เช่นในตัวอย่างข้างต้น ถ้าผู้ใช้พิมพ์ –f2, f3, f4,..,f13 จะแสดงข้อมูลว่างออกทางจอภาพ เมื่อต้องการข้อมูลในคอลัมภ์ที่ 2(คือเทอร์มินัล) จะต้องใช้ –f14

ตัวอย่างไฟล์ข้อมูลชื่อว่า cut.txt

vim is an advanced text editor that seeks to provide the power of the de-facto Unix editor 'Vi', with a more complete feature set.

cat cut.txt| cut -d' ' -f1-3

vim is an

คัดแยกคำจากผลลัพธ์ของคำสั่ง cat โดยแบ่งคำแต่ละคำด้วยช่องว่าง (-d' ') จากนั้นทำการตัดเอาเฉพาะคำที่อยู่ในฟิวด์ที่ 1 ถึง 3 คือคำว่า vim, is, an

cat cut.txt| cut -d' ' -f1,3,5,7,20

vim an text that a

คัดแยกคำจากผลลัพธ์ของคำสั่ง cat โดยแบ่งคำแต่ละคำด้วยช่องว่าง (-d' ') จากนั้นทำการตัดเอาเฉพาะคำที่อยู่ในฟิวด์ที่ 1,3, 5, 7 และ 20 เท่านั้น

cut -c 1,2 cut.txt

vi

คัดแยกคำจากไฟล์ชื่อว่า cut.txt โดยเลือกเอาเฉพาะอักษรตำแหน่งที่ 1(คืออักษร v) และ 2(คืออักษร i) มาแสดงผลเท่านั้น ถ้ากรณีที่ข้อมูลในไฟล์มีมากกว่า 1 บรรทัด จะแสดงอักษรตัวที่ 1 และ 2 ทุกๆ บรรทัด

cut -c 1-7 cut.txt

vim is

คัดแยกคำจากไฟล์ชื่อว่า cut.txt โดยเลือกเอาเฉพาะอักษรตำแหน่งที่ 1 ถึง 7 กรณีที่ข้อมูลในไฟล์มีมากกว่า 1 บรรทัด จะแสดงอักษรตัวที่ 1 ถึง 7 ทุกๆ บรรทัด

cut -c 1,5,7 cut.txt

คัดแยกคำจากไฟล์ชื่อว่า cut.txt โดยเลือกเอาเฉพาะอักษรตำแหน่งที่ 1, 5 และ 7 กรณีที่

ข้อมูลในไฟล์มีมากกว่า 1 บรรทัด จะแสดงอักษรตัวที่ 1, 5 และ 7 ทุกๆ บรรทัด

```
# cut -d ":" -f1 /etc/passwd
root
bin
daemon
adm
lp
sync
shutdown
...
```

คัดแยกคำจากไฟล์ชื่อ /etc/passwd โดยใช้อักษร delimeter คือ “:” แยกคำในไฟล์ข้อมูลออกจากกัน ผลจากคำสั่งดังกล่าวจะเลือกเอาเฉพาะคำในฟิวด์ที่ 1 ในทุกๆ บรรทัด

```
# cut -d ":" -f1,7 /etc/passwd
root:/bin/bash
bin:/sbin/nologin
daemon:/sbin/nologin
adm:/sbin/nologin
lp:/sbin/nologin
...
```

คัดแยกคำจากไฟล์ชื่อ /etc/passwd โดยใช้อักษร delimeter คือ “:” แยกคำในไฟล์ข้อมูลออกจากกัน ผลจากคำสั่งดังกล่าวจะเลือกเอาเฉพาะคำในฟิวด์ที่ 1 และ 7 ในทุกๆ บรรทัด

```
# cut -d ":" -f 1,6- /etc/passwd
root:/root:/bin/bash
bin:/bin:/sbin/nologin
daemon:/sbin:/sbin/nologin
adm:/var/adm:/sbin/nologin
lp:/var/spool/lpd:/sbin/nologin
…
```

คัดแยกคำจากไฟล์ชื่อ /etc/passwd โดยใช้อักษร delimeter คือ “:” แยกคำในไฟล์ข้อมูลออกจากกัน ผลจากคำสั่งดังกล่าวจะเลือกเอาเฉพาะคำในฟิวด์ที่ 1 และฟิวด์ตั้งแต่ฟิวด์ที่ 6 เป็นต้นไปมากแสดง ในทุกๆ บรรทัด

```
# cut -d ":" -f 2-5 --complement  /etc/passwd
root:/root:/bin/bash
bin:/bin:/sbin/nologin
daemon:/sbin:/sbin/nologin
```

ผลจากคำสั่งดังกล่าวจะเลือกเอาเฉพาะคำในฟิวด์ทั้งหมดยกเว้นฟิวด์ที่ 2-5(เนื่องจากใช้ออพชัน complement)

```
# w |cut -c1-12
22:11:11  up
USER     TTY
root      pts
root      pts
```

w คือคำสั่งแสดงผู้ใช้งานในระบบ ผลลัพธ์ที่ได้จาก w จะส่งต่อไปให้กับ cut คัดแยกเอาเฉพาะอักษรตั้งแต่ตัวที่ 1 ถึง 12 มาแสดงผล

ลองทดสอบ :

```
# cut -c -7 myfile.txt
# cut -f "1 2 3" -d : /etc/passwd
# cut -d : -f 5- file
# name=`who am i | cut -f1 -d' '`; echo $name
```

ดูคำสั่งอื่นประกอบ : *grep, paste*

คำสั่ง paste

File Text Manipulation : **paste**	
คำสั่ง	**paste** เป็นคำสั่งที่ใช้สำหรับเชื่อมต่อข้อมูลในไฟล์แบบบรรทัดต่อบรรทัด
Syntax	**paste *[options] files***
Example	**# paste x y z > file** **# ls | paste - - - -**

options:-

-d, --delimiters=*char* แยกคอลัมภ์แต่ละคอลัมภ์ด้วยอักษรที่กำหนด แทนการใช้ TAB

-s, --serial เชื่อมต่อข้อมูลในไฟล์ให้เป็นไฟล์เดียวกัน เมื่อมีมากกว่า 1 ไฟล์จะเชื่อมต่อข้อมูลในไฟล์แรกเสร็จก่อนจึงจะเอาข้อมูลในไฟล์ที่ 2 มาเชื่อมต่อทีหลัง

-d, --delimiters=LIST กำหนดตัวอักษรสำหรับใช้เป็น delimiter

-s, --serial ผสานไฟล์ในแต่ละบรรทัดของแต่ละไฟล์เป็นบรรทัดเดียวกัน

- จัดเรียงจำนวนคอลัมภ์ตามที่ผู้ใช้ต้องการ

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

```
# paste x y z > file
```

ทำการรวมข้อมูลในไฟล์ x y และ z เข้าด้วยกัน โดยแบ่งออกเป็น 3 คอลัมภ์ ข้อมูลคอลัมภ์แรกเป็นของไฟล์ x คอลัมภ์ที่สองเป็นของ y และคอลัมภ์สุดท้ายเป็นของ z

```
# ls | paste - -
```

แสดงข้อมูลที่ได้จาก ls เป็น 2 คอลัมภ์

```
# paste test.txt > test1.txt
```

แทนที่ข้อมูลใน test1.txt ด้วยข้อมูลของ test.txt

ตัวอย่างข้อมูล

f1.txt	f2.txt
a	1
b	2
c	

```
# paste f1.txt f2.txt
```

```
a       1
b       2
c
```

จากข้อมูลในตารางข้างบน เมื่อออกคำสั่ง paste f1.txt f2.txt ผลที่ได้รับคือจะทำการรวมข้อมูลของ f1.txt และ f2.txt เป็นไฟล์เดียวกันและเรียงเป็น 2 คอลัมภ์ ข้อมูลของ f1.txt จะเป็นคอลัมภ์ที่ 1

paste f2.txt f1.txt

```
1       a
2       b
        c
```

จากข้อมูลในตารางข้างบน เมื่อออกคำสั่ง paste f2.txt f1.txt ผลที่ได้รับคือจะทำการรวมข้อมูลของ f2.txt และ f1.txt เป็นไฟล์เดียวกันและเรียงเป็น 2 คอลัมภ์ ข้อมูลของ f2.txt จะเป็นคอลัมภ์ที่ 1

paste -s -d : f1.txt f2.txt

```
a:b:c
1:2
```

ทำการเชื่อมต่อไฟล์ด้วย option –s (past จะเชื่อมข้อมูลด้วย : ในไฟล์ f1.txt ให้เสร็จก่อน ในที่นี้คือ a:b:c จากนั้นจึงไปเชื่อมกับไฟล์ f2.txt), -d : เป็นการระบุตัวคั่นในแต่ละคำที่เชื่อมต่อ ผลที่ได้คือ ทำการเชื่อมข้อมูลในไฟล์ f1.txt เข้าด้วยกันและคั่นด้วย : เมื่อเชื่อมต่อ f1.txt เสร็จก็ทำการเชื่อมต่อ f2.txt ต่อท้ายไฟล์

ตัวอย่างข้อมูล

fa.txt	fb.txt	fc.txt
20	60	70
60	90	90
90	12	80
12	14	12

paste -d"+-" fa.txt fb.txt fc.txt

```
20+60-70
60+90-90
90+12-80
12+14-12
```

เชื่อมต่อไฟล์ fa, fb และ fc โดยคั่นด้วย + และ -

ตัวอย่างการใช้งาน(ต่อ) :

File1	File2	File3
Somsri	1234	Somsri@hotmail.com
Manee	5678	Manee@hotmail.com
Suwit	9101	Suwit@hotmail.com
Narong	1213	Narong@hotmail.com
Dokya	1415	Dokya@hotmail.com

```
# paste file1 file2
Somsri  1234
Manee   5678
Suwit   9101
Narong  1213
Dokya   1415
```

จากตัวอย่าง คำสั่ง paste จะนำข้อมูลของแต่ละแถวในไฟล์ file1 และ file2 มาเชื่อมต่อกัน ผลลัพธ์คือ ข้อมูลในแถวแรกของไฟล์ file1 จะเชื่อมกับแถวแรกของไฟล์ file2 และแสดงผลลัพธ์ออกทางหน้าจอภาพ ในกรณีที่สลับไฟล์ระหว่าง file1 และ file2 จะให้ผลลัพธ์ดังนี้

```
1234    Somsri
5678    Manee
9101    Suwit
1213    Narong
1415    Dokya
# paste file2 file1 > result
```

เหมือนตัวอย่างข้างต้น แต่ผลลัพธ์ที่ได้นำไปเก็บไว้ในไฟล์ชื่อว่า result แทน

```
# paste -s -d: file1 file2
Somsri:Manee:Suwit:Narong:Dokya:
1234:5678:9101:1213:1415:
```

ทำการเชื่อมต่อไฟล์ข้อมูล file1 และ file2 เข้าด้วยกัน แบบต่อเนื่องเป็นแถวเดียว(serial) โดยใช้ออพชัน –s และคั่นแต่ละคำด้วย delimiter (ตัวอักษร :) โดยใช้ออพชัน -d

```
# ls /bin/ | paste - - - -
alsacard      alsaunmute    arch    awk
basename      bash          cat     chgrp
chmod         chown         cp      cpio
csh           cut           date    dbus-cleanup-sockets
```

แสดงรายการในไดเรคทรอรี /bin ด้วยคำสั่ง ls จากนั้นส่งผลลัพธ์ให้กับคำสั่ง paste ซึ่งคำสั่ง paste จะจัดเรียงการแสดงผลใหม่เป็นจำนวนคอลัมภ์เท่ากับ 4 คอลัมภ์ (ใช้อักษร – แทนจำนวนคอลัมภ์)

```
# paste -d"+-" file1 file2 file3
Somsri+1234-Somsri@hotmail.com
Manee+5678-Manee@hotmail.com
Suwit+9101-Suwit@hotmail.com
Narong+1213-Narong@hotmail.com
Dokya+1415-Dokya@hotmail.com
```

ทำการเชื่อมต่อไฟล์ข้อมูล file1, file2 และ file3 เข้าด้วยกัน โดยใช้ delimiter “+” คั่นระหว่างไฟล์ fil1 และ file2 ผลที่ได้จะนำไปเชื่อมต่อกับ file3 โดยใช้ delimiter “-” คั่นไว้

```
# paste -s -d"\t\n" file1 file2
Somsri  Manee
```

```
Suwit  Narong
Dokya

1234   5678
9101   1213
1415
```
ทำการเชื่อมต่อไฟล์ข้อมูล file1 และ file2 เข้าด้วยกัน โดยคั่นแต่ละคำด้วย delimiter แทบ(\t) และขึ้นบรรทัดใหม่(\n)

ลองทดสอบ :

```
# paste -d '+' file1 file2
# paste fa.txt fb.txt fc.txt | awk '{print "("$1"+"$2"-"$3")="$1+$2-$3}'
# paste -s -d"\t\n" list
# paste  -d"!@" file1 file2 file3 > result
# ls | paste  -d"\t\t\t\n" -s -
```

ดูคำสั่งอื่นประกอบ : *cut, grep, pr*

คำสั่ง tr

File Text Manipulation : **tr**	
คำสั่ง	**tr** เป็นคำสั่งที่ใช้สำหรับแปลงอักษรหรือข้อความ
Syntax	**tr [*options*] *files***
Example	**# tr**

options:-

-c, -C, --complement เปลี่ยนตัวอักษรที่ต้องการ เป็นรหัส ascii ตั้งแต่ 001-337

-d, --delete ลบตัวอักษรที่ต้องการออก โดยไม่ให้มีการแสดงผล

-s, --squeeze-repeats แทนที่ตัวอักษรที่ต้องการในไฟล์ข้อมูล

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

ตัวอย่างไฟล์ trfile.txt

```
Vim is an advanced text editor that seeks to provide the
power of the de-facto Unix editor 'Vi', with a more
complete feature set. It's useful whether you're already
# cat trfile.txt | tr 'A-Z' 'a-z'
vim is an advanced text editor that seeks to provide the
power of the de-facto unix editor 'vi', with a more
complete feature set. it's useful whether you're already
```

จากตัวอย่าง คำสั่ง tr จะทำการแปลงข้อมูลในไฟล์ trfile.txt เป็นอักษรตัวเล็กทั้งหมด (แปลงอักษรตัวใหญ่ 'A-Z' เป็นอักษรตัวเล็ก 'a-z')

```
# echo "12345678 9247" | tr 123456789 computerh
```

computer hope

คำสั่ง echo จะพิมพ์ข้อความ “12345678 9247” และส่งข้อความดังกล่าวให้กับคำสั่ง tr ผ่านไปป์ เมื่อคำสั่ง tr ทำงาน จะทำการแปลงตัวอักษร 1 เป็น c, 2 แปลงเป็น o, 3 แปลงเป็น m, .., 8 แปลงเป็น r ตามลำดับ เมื่อไปถึงอักษร 9 จะถูกแทนที่ด้วยตัวอักษรว่าง ต่อจากนั้นอักษร 9 จะถูกแทนที่ด้วยอักษร h, 2 แทนด้วย o, 4 แทนด้วย p และ 7 แทนด้วย e ดังรูปที่ 5-11

123456789
Computerh

12345678 9247
Computer hope

รูปที่ 5-11 แสดงการแปลงตัวอักษรด้วยคำสั่ง tr

ตัวอย่างไฟล์ file

```
Somsri@hotmail.com
Manee@hotmail.com
Suwit@hotmail.com
Narong@hotmail.com
Dokya@hotmail.com
# tr -d @ < file > new.file
Somsrihotmail.com
Maneehotmail.com
Suwithotmail.com
Naronghotmail.com
Dokyahotmail.com
```

ลบตัวอักษร @ ออกจากไฟล์ file และเก็บผลลัพธ์ไว้ในไฟล์ใหม่ชื่อว่า new.file (ตัวอักษร < หมายถึงการนำเข้าข้อมูลเพื่อประมวลผล และ > หมายถึงการส่งออกข้อมูล ดูรายละเอียดเพิ่มเติมได้จากคำสั่ง การเปลี่ยนทิศทางหรือ redirection)

```
# tr -d '\0' < textfile > newfile
```

ลบตัวอัการ NULL ออกจากไฟล์ textfile และเก็บผลลัพธ์ไว้ในไฟล์ใหม่ชื่อว่า new.file

```
# tr -s '[:space:]' '[\:*]' < trfile.txt > new.file
Vim:is:an:advanced:text:editor:that:seeks:to:provide:the:power:of:the:de-
facto:Unix:editor:
```

แทนที่ตัวอักษรว่าง(space) ในไฟล์ด้วยตัวอักษร “:” ทั้งไฟล์ข้อมูล ผลลัพธ์เก็บไว้ในไฟล์ new.file

```
# tr -s '\n' < textfile > newfile
```

แทนที่คำสั่งขึ้นบรรทัดใหม่ในไฟล์ข้อมูลที่มีมากกว่า 1 บรรทัดด้วยคำสั่งขึ้นบรรทัดใหม่ 1 บรรทัดเท่านั้น ผลลัพธ์เก็บไว้ในไฟล์ newfile

ลองทดสอบ :

```
# echo "a test" | tr t p
```

```
# echo "a test" | tr aest 1234
# echo "a test" | tr -d t
# echo "a test" | tr '[:lower:]' '[:upper:]'
# cat columns.txt | tr '[a-z]' '[A-Z]'
# cat columns.txt | tr '[a-z]' '[A-Z]' > UpCaseColumns.txt
# echo 'linux' | tr "[:lower:]" "[:upper:]"
# echo 'linux' | tr "a-z" "A-Z"
# echo 'I LovE linuX. one is better Than 2' | tr "a-z" "A-Z"
```

ดูคำสั่งอื่นประกอบ : *ed, sed, sh*

คำสั่ง sort

File Text Manipulation : **sort**	
คำสั่ง	**sort** เป็นคำสั่งที่ใช้สำหรับทำการจัดเรียงข้อมูลในไฟล์ตามลำดับ (ทั้งนี้จะถือว่าข้อมูลแต่ละบรรทัดเป็น 1 record และจะใช้ field แรกเป็น key)
Syntax	**sort *[options] [files]***
Example	**# sort data.txt** **# sort -r data.txt**

options:-

-b, --ignore-leading-blanks ไม่นำอักษรว่างและแทบ(Tab) มาประมวลผลร่วมด้วย

-c, --check ตรวจสอบไฟล์ว่ามีการเรียงลำดับแล้วหรือไม่ ถ้าเรียงแล้วจะไม่แสดงผลออกมา

-d, --dictionary-order เรียงลำดับตัวอักษรตามที่ปรากฏในดิกชันนารี

-f, --ignore-case ไม่ต้องสนใจว่าอักษรที่ใช้เป็นเล็กหรือใหญ่ จะประมวลผลเหมือนกัน

-g, --general-numeric-sort เรียงตามลำดับตัวเลขแทนตัวอักษร

-i, --ignore-nonprinting ไม่ต้องประมวลผลอักษรที่ไม่สามารถแสดงผลได้ เช่น รหัสขึ้นบรรทัดใหม่ กลุ่มคำสั่งที่ใช้ในการควบคุม เช่น line feed เป็นต้น

-k, --key=POS1[,POS2] ประมวลผลตำแหน่งที่กำหนดใน POST1, POST2 (POST1= ตำแหน่งเริ่มต้น, POST2=ตำแหน่งที่สิ้นสุด)

-n เรียงลำดับโดยพิจารณาจาก aritmatic

-ofile, --output=file แสดงผลออกไว้ยังไฟล์

-m, --merge รวมไฟล์เข้าด้วยกันเหลังจากมีการเรียงลำดับแล้ว

-r, --reverse สลับลำดับตัวอักษรจาก z ไป a

ตัวอย่างการใช้งาน :

sort file.txt

จัดเรียงข้อมูลตามลำดับตัวอักษรใน file.txt

sort -r file.txt

จัดเรียงข้อมูลแบบกลับลำดับตัวอักษรใน file.txt

```
# ls -al | sort -n -k5
-rw-------  1 root   root     0 Nov 15 18:14 .Xauthority
-rw-------  1 root   root     5 Jan 16 11:29 date.txt.lock
-rw-r--r--  1 root   root    21 Jan 12 11:59 date.txt
-rw-r--r--  1 root   root    24 Jan  6  2007 .bash_logout
-rw-------  1 root   root    26 Nov  8 16:15 .dmrc
-rw-------  1 root   root    35 Jan 19 11:10 .lesshst
-rw-r--r--  1 root   root    38 Nov 22 09:50 error
-rw-r--r--  1 root   root    81 Nov  8 16:16 .gtkrc-1.2-gnome2
-rw-r--r--  1 root   root    82 Nov 22 11:02 install.log
```

ทำการจัดเรียงข้อมูลที่ได้จากคำสั่ง ls –al โดยเรียงตามขนาดของไฟล์ (-k5 ใช้ระบุว่าเป็นคอลัมภ์ที่ 5) เรียงจากมากไปน้อย

```
# sort files > files.sorted
```

จัดเรียงข้อมูลใน files แล้วนำผลที่ได้จากการเรียงลำดับไปเก็บใน files.sorted

```
# sort file | less
```

จัดเรียงข้อมูลใน file แล้วนำผลที่ได้จากการเรียงลำดับไปแสดงผลด้วย less อีกครั้งเพื่อให้ง่ายต่อการดูข้อมูล

```
# sort -o outfile infile
```

จัดเรียงข้อมูลใน infile แล้วนำผลที่ได้จากการเรียงลำดับไปเก็บใน outfile

ลองทดสอบ :

```
# ps auxw | sort
# ls -al | sort +4n | more
# du -s * | sort -nr
```

ดูคำสั่งอื่นประกอบ : *commm, join, uniq*

คำสั่ง uniq

File Text Manipulation : **uniq**	
คำสั่ง	**uniq** เป็นคำสั่งที่ใช้สำหรับรวมบรรทัดที่เหมือนกันในไฟล์ข้อมูล
Syntax	**uniq [*options*] [*file1* [*file2*]]**
Example	**# uniq list list.new** **# uniq -d myduplicate**

options:-

-d, --repeated แสดงบรรทัดที่ซ้ำกันในไฟล์ออกมาเพียง 1 บรรทัดเท่านั้น

-u, --unique แสดงผลข้อความที่ไม่ซ้ำกัน

-c, --count นับจำนวนบรรทัดที่ซ้ำ

ตัวอย่างการใช้งาน :

```
# uniq list list.new
```

รวมบรรทัดที่เหมือนกันในไฟล์ list ให้เหลือเพียงบรรทัดเดียวและเก็บไว้ในไฟล์ list.new

sort list | uniq -d

จัดเรียงข้อมูลด้วยคำสั่ง sort ในไฟล์ชื่อ list ผลลัพธ์ที่ได้ส่งให้กับคำสั่ง uniq เพื่อรวมบรรทัดให้เหลือข้อมูลที่ไม่ซ้ำกัน

ตัวอย่างไฟล์ชื่อว่า my.books

```
Atopic Dermatitis for Dummies
Atopic Dermatitis for Dummies
Chronic Rhinitis Unleashed
Chronic Rhinitis Unleashed
Chronic Rhinitis Unleashed
Learn Nasal Endoscopy in 21 Days
```

uniq my.books

```
Atopic Dermatitis for Dummies
Chronic Rhinitis Unleashed
Learn Nasal Endoscopy in 21 Days
```

แสงผลข้อความที่ซ้ำกันเพียงบรรทัดเดียว พร้อมกับแสดงข้อความที่ไม่ซ้ำกันด้วย

uniq -u my.books

```
Learn Nasal Endoscopy in 21 Days
```

แสงผลข้อความที่ไม่ซ้ำกัน

uniq -d my.books

```
Atopic Dermatitis for Dummies
Chronic Rhinitis Unleashed
```

แสงผลข้อความที่ซ้ำกันเพียงบรรทัดเดียวเท่านั้น

uniq -c my.books

```
2 Atopic Dermatitis for Dummies
3 Chronic Rhinitis Unleashed
1 Learn Nasal Endoscopy in 21 Days
```

นับจำนวนข้อความที่ซ้ำกันและแสดงผลจำนวนนับไว้ต้นบรรทัด

ลองทดสอบ :

uniq myfile1.txt > myfile2.txt

ดูคำสั่งอื่นประกอบ : *comm, pack, pcat, sort, uncompress*

คำสั่ง tee

File Text Manipulation : **tee**	
คำสั่ง	**tee** เป็นคำสั่งที่ใช้สำหรับสำเนาข้อความออกทางไฟล์และ stdout พร้อมๆ กัน
Syntax	**tee [*options*] *files***
Example	**# ls -l | tee savefile**

options:-

-a, --append เขียนข้อมูลต่อท้ายไฟล์โดยไม่มีการเขียนทับข้อมูลในไฟล์เดิม

-i, --ignore-interrupts ไม่สนใจเมื่อเกิดการขัดจังหวะ(เช่น ในกรณีที่สั่งงานคำสั่ง tee ไปแล้วและมีการกดปุ่ม delete คำสั่ง tee จะไม่สนใจการขัดจังหวะดังกล่าว)

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

ls -l | tee savefile

สำเนาข้อความผลลัพธ์ที่เกิดจากคำสั่ง ls –l ไปเก็บไว้ยังไฟล์ชื่อว่า savefile

ls *.txt | wc -l | tee /dev/tty count.txt

```
13
13
```

ผลลัพธ์ที่ได้จากคำสั่ง ls *.txt คือรายชื่อของไฟล์ทุกไฟล์ที่มีส่วนขยายเป็น .txt ต่อจากนั้นส่งผลลัพธ์ดังกล่าวมาให้คำสั่ง wc นับจำนวนไฟล์ จำนวนที่นับได้จะถูกส่งต่อไปให้กับคำสั่ง tee โดยคำสั่งดังกล่าวจะทำงาน 2 อย่างคือ ส่งจำนวนนับของไฟล์ไปแสดงผลที่หน้าจอภาพ(/dev/tty) และส่งผลดังกล่าวไปเก็บไว้ยังไฟล์ชื่อว่า count.txt

date | tee file1 file2

```
Sat Feb  6 06:26:35 ICT 2010
```

ส่งผลลัพธ์จากคำสั่ง date ไปเก็บไว้ในไฟล์ file1 และ file2

uptime | tee -a file2

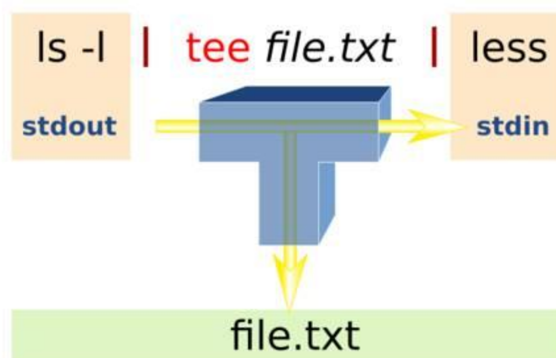
```
Sat Feb  6 06:26:35 ICT 2010
 06:27:44 up  2:49,  1 user,  load average: 0.00, 0.00, 0.00
```

ส่งผลลัพธ์จากคำสั่ง uptime ไปเก็บไว้ในไฟล์ file2 แบบเขียนต่อท้ายไฟล์

ps -ax | tee processes.txt | more

```
Atopic Dermatitis for Dummies
Chronic Rhinitis Unleashed
```

ส่งผลลัพธ์จากคำสั่ง ps -ax ซึ่งเป็นคำสั่งที่แสดงโพรเซสที่กำลังทำงานในระบบ ไปเก็บไว้ในไฟล์ processes.txt พร้อมกับส่งไปแสดงผลกับคำสั่ง more ด้วย



รูปที่ 5-12 แสดงการทำงานของคำสั่ง tee

ลองทดสอบ :

$ ls | tee file1 file2 file3

ดูคำสั่งอื่นประกอบ : *cat*

คำสั่ง echo

File Text Manipulation : **echo**	
คำสั่ง	**echo** เป็นคำสั่งที่ใช้สำหรับแสดงข้อความออกทาง standard output (จอภาพ)
Syntax	**echo *[options] [string]***
Example	**# echo Hello world** **# echo 'Today is ' $(date)**

options:-

-e อนุญาติให้สัญลักษณ์ที่อยู่หลังเครื่องหมาย \ ทำงาน

-E ไม่อนุญาติให้สัญลักษณ์ที่อยู่หลังเครื่องหมาย \ ทำงาน

-n ไม่พิมพ์รหัสการขึ้นบรรทัดใหม่

\a สั่งให้ส่งเสียง bell

\b สั่งพิมพ์ backspace

\f สั่งพิมพ์ form feed

\n สั่งขึ้นบรรทัดใหม่

\r Carriage return

\t Horizontal tab

\v Vertical tab

\\ สั่งพิมพ์ \

ตัวอย่างการใช้งาน :

echo Hello world

แสดงข้อความ Hello world ออกจอภาพ

echo *

แสดงรายชื่อไฟล์ทั้งหมดในไดเรคทรอรีปัจจุบัน ออกจอภาพ

echo * | wc

ข้อมูลรายชื่อไฟล์ทั้งหมดในไดเรคทรอรีปัจจุบันที่ได้จาก echo * จะถูกส่งต่อไปให้ wc เพื่อทำการนับจำนวนคำ และแสดงออกจอภาพ

echo -e "Warning: ringing bell \a"

แสดงข้อความ Warning: ringing bell ออกจะภาพ พร้อมกับเสียง bell

echo 'Welcome'

แสดงข้อความ Welcome ออกจอภาพ

echo 'File has been deleted 1' > /tmp/log.txt

เพิ่มข้อความ File has been deleted 1 ไปยังไฟล์ชื่อว่า log.txt

echo 'File has been deleted 2' >> /tmp/log.txt

เพิ่มข้อความ File has been deleted 2 ต่อท้ายไฟล์ชื่อว่า log.txt

echo "Today's date is $(date)"

Today's date is Tue Jan 19 11:22:35 ICT 2010

แสดงข้อความ Today's date is ต่อด้วยวันที่ ที่ได้จากคำสั่ง $(date) ออกจอภาพ

```
# myvar='This is my environment variable!'
# echo $myvar
This is my environment variable!
```

แสดงข้อความ This is my environment variable! ออกจอภาพ

```
# echo -e "Country \bThai \bIndia"
```

ลบ TAB ออกจากข้อความ ส่งผลให้ข้อความที่แสดงออกมาคือ CountryThaiIndia ออกจอภาพ

```
# echo -e "Country\tThai\tIndia"
```

ใส่ TAB ในข้อความ ส่งผลให้ข้อความที่แสดงออกมาคือ Country Thai India ออกจอภาพ

```
# echo -en $"Suppress printing of newline after text."
Suppress printing of newline after text. #
```

แสดงข้อความ Suppress printing of newline after text ออกจอภาพ โดยไม่ขึ้นบรรทัดใหม่

```
# echo -e "\n Projects: \n\n\tplan \n\tcode \n\ttest\n"

 Projects:

        plan
        code
        test
```

เริ่มจากการขึ้นบรรทัดใหม่ 1 ครั้ง(\n) พิมพ์คำว่า Projects ขึ้นบรรทัดใหม่อีก 2 ครั้ง จากนั้นพิมพ์แทบ(\t) ต่อด้วย plan ตามลำดับ

ลองทดสอบ :

```
# echo "yes"
# echo "Today's date is $(date)" > /tmp/date.txt
# echo $(ls -l)
# echo "Today's date is $(date)" > /tmp/date.txt
# echo $PATH
# echo My name is $USER - Home directory=$HOME
# echo $LOGNAME
# echo $PATH
```

ดูคำสั่งอื่นประกอบ : *printf, tr*

คำสั่ง sdiff

File Text Manipulation : **sdiff**	
คำสั่ง	**sdiff** เป็นคำสั่งที่ใช้สำหรับเปรียบเทียบไฟล์ 2 ไฟล์แบบบรรทัดต่อบรรทัด พร้อมกับเชื่อมไฟล์ให้ด้วย

Syntax	***sdiff -o outfile [options] from to***
Example	**# sdiff myfile.txt myfile2.txt**

options:-

ตัวอย่างการใช้งาน :

> **# sdiff file1 file2**
> ค้นหาความแตกต่างระหว่าง file1 และ file2 จากนั้นทำการรวมไฟล์ทั้งสองเข้าด้วยกัน คล้ายกับการทำงานของ diff

ลองทดสอบ :

> **# diff paper_v1 paper_v2**

ดูคำสั่งอื่นประกอบ : *diff, ed*

คำสั่ง sed

File Text Manipulation : **sed**	
คำสั่ง	**sed** เป็นคำสั่งที่ใช้สำหรับแก้ไขไฟล์ข้อความได้แบบอัตโนมัติ เช่น การลบบรรทัด การแทนที่ข้อความ หรือการแทรกบรรทัด เป็นต้น
Syntax	***sed [OPTION]... {script-only-if-no-other-script} [input-file]...***
Example	**# sed = myfile.txt | sed 'N;s/\n/\. /'**

options:-

-e เขียน script หรือเขียนคำสั่ง

-f ระบุไฟล์สคริปต์ที่สร้างขึ้นใช้ในกรณีที่ เขียนสคริปต์ไว้แล้ว

-n แสดงผลข้อมูล

ตัวอย่างการใช้งาน :

ตัวอย่างข้อมูลใน file1.txt

```
hscripts has many valuable free scripts
It is the parent site of www.forums.hscripts.com
hscripts include free tutorials and free gif images
free DNS lookup tool
Purchase scripts from us
A webmaster/web master resource website
```

sed G file1.txt>file2.txt

```
hscripts has many valuable free scripts

It is the parent site of www.forums.hscripts.com

hscripts include free tutorials and free gif images
```

Option G(double space) จะใส่บรรทัดว่างระหว่างบรรทัด เป็นเท่าตัว ในตัวอย่าง เดิมแต่ละแถวห่างกัน 1 บรรทัดว่าง เมื่อใช้ G จะส่งผลให้ข้อมูลใน file2.txt มีจำนวนบรรทัด

ว่างในแต่ละแถวเป็นเท่าตัวคือ 2 บรรทัด

```
# sed = file1.txt | sed 'N;s/\n/\. /'
1. hscripts has many valuable free scripts
2. It is the parent site of www.forums.hscripts.com
3. hscripts include free tutorials and free gif images
...
```

จากตัวอย่างข้างบนจะทำการแทนค่าหมายเลขบรรทัด ตามด้วยจุด เข้าไปยังส่วนหัวของแต่ละบรรทัดของข้อมูลใน file1.txt

sed 's/scripts/javascript/g' file1.txt

ทำการค้นหาคำว่า scripts ใน file1.txt เมื่อคนหาเจอจะแทนที่ด้วย คำว่า javascript

```
# sed -n '$=' file1.txt
6
```

นับจำนวนของแถวใน file1.txt และแสดงผล

sed 's/string1/string2/g' example.txt

แทนที่คำว่า string2 ด้วยคำว่า string1 ในไฟล์ example.txt

sed -e '1d' example.txt

ลบบรรทัดที่ว่างในไฟล์ example.txt ออก

sed '/ *#/d; /^$/d' example.txt

ลบบรรทัดว่างและ comment ในไฟล์ example.txt ออก

sed -e '1d' example.txt

ตัดบรรทัดแรกของในไฟล์ example.txt ทิ้ง 1 บรรทัด

sed -n '/string1/p' example.txt

แสดงเฉพาะบรรทัดที่มีคำว่า string1 ในไฟล์ example.txt เท่านั้น

sed -e 's/ *$//' example.txt

ลบอักษรหรือข้อความที่ว่างส่วนท้ายของทุกๆ บรรทัด ในไฟล์ example.txt ออกทั้งหมด

sed -e 's/string1//g' example.txt

ลบเฉพาะข้อความที่มีชื่อ string1 ในไฟล์ example.txt ออกเท่านั้นโดยไม่แก้ไขข้อความอื่นๆ ทั้งสิ้น

sed -n '1,5p' example.txt

แสดงเฉพาะบรรทัดที่ 1 ถึงบรรทัดที่ 5 ในไฟล์ example.txt เท่านั้น

sed -n '5p;5q' example.tx

แสดงเฉพาะบรรทัดที่ 5 ในไฟล์ example.txt เท่านั้น

sed -e 's/00*/0/g' example.txt

ในกรณีที่ไฟล์ example.txt มี 0 ติดกันหลายตัว ให้ทำการลบทิ้งแล้วเหลือไว้เพียง 1 ตัวเท่านั้น

ลองทดสอบ :

```
# sed -e '/^$/d' your_file.txt
sed '/^$/d' example.txt
sed '/ *#/d; /^$/d' example.txt
```

ดูคำสั่งอื่นประกอบ : *awk, ed, grep*

5.3.7 กลุ่มคำสั่งเกี่ยวกับ File Compression

คำสั่ง gzip, gunzip

File Compression : **gzip, gunzip**	
คำสั่ง	**gzip** เป็นคำสั่งที่ใช้สำหรับบีบอัดไฟล์ข้อมูลที่มีส่วนขยาย .gz **gunzip** เป็นคำสั่งที่ใช้สำหรับคลายไฟล์ข้อมูลที่มีส่วนขยาย .gz
Syntax	***gzip [options] [files]*** ***gunzip [options] [files]*** ***zcat [options] [files]***
Example	**# gzip myfile** **# gunzip myfile.gz**

options:-

-*n*, --fast, --best กำหนดความเร็วในการบีบอัดไฟล์ เมื่อกำหนดเป็น fast(-1) จะมีความเร็วสูงสุด แต่ขนาดของไฟล์จะไม่เล็กที่สุด เมื่อกำหนด best(-9) จะบีบข้อมูลได้มากที่สุดแต่จะใช้เวลามาก ค่า default ในกรณีที่ไม่ได้กำหนดไว้คือ -6

-c, --stdout, --to-stdout แสดงผลไปยัง standard output โดยไม่มีการเปลี่ยนแปลงไฟล์ต้นฉบับ

-d, --decompress, --uncompress คลายไฟล์ที่บีบอัดไว้

-f, --force ให้ทำการบีบอัด แม้ว่าจะเกิดข้อผิดพลาดก็ตาม

-h, --help คำสั่งช่วยเหลือ

-l, --list แสดงรายการไฟล์ที่บีบอัดไว้

-r –recursive ทำการบีบอัดไดเรคทรอรีย่อยทั้งหมด

-t, --test ทดสอบการบีบอัดเพื่อตรวจสอบความถูกต้องของข้อมูล โดยไม่มีการบีบอัดจริงๆ

-v, --verbose แสดงการบีบอัดไฟล์และเปอร์เซ็นในการบีบอัด

ตัวอย่างการใช้งาน :

gzip myfile
myfile.gz
บีบอัดข้อมูลไฟล์ชื่อ myfile ไฟล์ที่ถูกบีบอัดแล้วจะมีส่วนขยายเป็น .gz อัตโนมัติ และไฟล์ต้นฉบับจะถูกลบออกไปด้วย

gunzip -f myfile.gz
คลายไฟล์ที่บีบอัดข้อมูลไว้ชื่อ myfile.gz ถ้ามีไฟล์อยู่แล้วจะทำการเขียนทับ(-f) ทันทีและลบไฟล์ต้นฉบับทิ้งด้วย(myfile.gz)

gzip -9 myfile
บีบอัดข้อมูลไฟล์ชื่อ myfile เมื่อใช้ -9 การบีบอัดจะใช้เวลานานมากแต่ไฟล์ที่บีบอัดจะมี

คุณภาพสูง

```
# gzip -r somedir
```
บีบอัดข้อมูลทุกไฟล์ในไดเรคทรอรี somedir แต่ไม่บีบอัดไดเรคทรอรี somedir

```
# gunzip -r somedir
```
คลายไฟล์ที่บีบอัดไว้ทุกไฟล์ในไดเรคทรอรี somedir

```
# gzip -c file1 > foo.gz
```
บีบอัดข้อมูลไฟล์ file1 โดยใช้ option –c เพื่อเปลี่ยนจาก standard output คือจอภาพไปยังไฟล์ชื่อว่า foo.gz แทน

```
# gzip -c file1 file2 > foo.gz
```
บีบอัดข้อมูลไฟล์ file1 และ file2 พร้อมกันโดยใช้ option –c และเปลี่ยนจาก standard output คือจอภาพไปยังไฟล์ชื่อว่า foo.gz แทน

ลองทดสอบ :

```
# gunzip -c something.tar.gz | tar xvf -
# gzip -cd old.gz | gzip > new.gz
# cat file1 file2 | gzip > foo.gz
```

ดูคำสั่งอื่นประกอบ : *cat, compress, pack, tar, uncompress*

คำสั่ง bzip2, bunzip2

File Compression : **bzip2, bunzip2**	
คำสั่ง	**bzip2** เป็นคำสั่งที่ใช้สำหรับบีบอัดข้อมูล ที่มีส่วนขยายเป็น .bz2 **bunzip2** เป็นคำสั่งที่ใช้สำหรับคลายการบีบอัดข้อมูลแบบชนิด bzip2
Syntax	**bzip2 *[options] filenames*** **bunzip2 *[options] filenames***
Example	**# bunzip2 filename** **# tar -cjf manee.tar.bz2 /home/manee**

options:-

-c –stdout ส่งผลลัพธ์จากการบีบอัดหรือคลาย ไปยัง standart output เช่น จอภาพ หรือไฟล์

-d –decompress คลายข้อมูลที่มีการบีบอัดไว้

-f –force เขียนทับไฟล์เดิมที่อยู่ในระบบ

-h –help แสดงข้อความช่วยเหลือ

-k –keep ไม่ลบไฟล์ต้นฉบับที่สั่งให้บีบอัด

-q –quiet ไม่แสดงผลที่ผิดพลาดออกมา

-t –test ทดสอบการบีบอัดก่อนการบีบจริงเพื่อตรวจสอบความผิดพลาดก่อน

-v –verbose แสดงข้อความการบีบอัด

-1 .. -9 กำหนดขนาดไฟล์ที่ต้องการบีบอัด มีหน่วยเป็น kilobyte เช่น 100k คือ -1

-z –compress สั่งบีบอัดข้อมูล

--fast ทำการบีบอัดข้อมูลแบบเร็ว

--best ทำการบีบอัดข้อมูลคุณภาพสูง แต่จะใช้เวลานานกว่าแบบ --fast

ตัวอย่างการใช้งาน :

bzip2 a.txt b.txt
-rw-r--r-- 1 root root 14 Jan 18 07:46 a.txt.bz2
-rw-r--r-- 1 root root 14 Jan 18 07:46 b.txt.bz2
ทำการบีบอัดไฟล์ 2 ไฟล์พร้อมกัน(ด้วย bzip2) ส่งผลให้ ได้ไฟล์ a.bz2 และ b.bz2 พร้อมกับลบไฟล์ต้นฉบับคือ a.txt และ b.txt ด้วย

bunzip2 a.bz2 b.bz2
คลายการบีบอัดไฟล์ 2 ไฟล์พร้อมกัน(ด้วย bunzip2) ส่งผลให้ ได้ไฟล์ a.txt และ b.txt พร้อมกับลบไฟล์ต้นฉบับคือ a.bz2 และ b.bz2 ทิ้งไป

bzip2 -c a.txt b.txt > ab.bz2
ทำการบีบอัดไฟล์ 2 ไฟล์พร้อมกัน(ด้วย bzip2) ส่งผลให้ ได้ไฟล์ ab.bz2 เนื่องจากใช้ –c ทำให้เปลี่ยนผลลัพธ์จากการบีบอัดไว้ในไฟล์ชื่อ ab.bz2 แทน ไฟล์ต้นฉบับจะไม่ถูกลบ

tar -cjf mana.tar.bz2 /home/username/mana
ทำการบีบอัดไดเรคทรอรี /home/username/mana ส่งผลให้ ได้ไฟล์ชื่อ mana.tar.bz2 (option –j ใช้เหมือนกับ tar)

bzip2 -c -1 ox.txt > ox.txt.bz2
ทำการบีบอัดไฟล์ชื่อว่า ox.txt ผลลัพธ์ที่ได้จะมีชื่อไฟล์ว่า ox.txt.bz2 และมีขนาดไม่เกิน 100 KB

tar -cf allfile.tar file1 file2 file3; bzip2 allfile.tar
ทำการสำรองไฟล์หรือรวมไฟล์ คือ file1, file2, file3 เป็นชื่อ allfile.tar ด้วยคำสั่ง tar จากนั้นบีบอัดไฟล์ด้วย bzip2 ไฟล์ allfile.tar เป็น allfile.tar.bzip2 อีกครั้ง(; คือการสั่งให้เชลล์ทำงานมากกว่า 1 คำสั่ง แต่คำสั่งแรกต้องทำเสร็จก่อนจึงจะทำคำสั่งที่ 2 ต่อได้)

tar -cjf allfile.tar.bz2 file1 file2 file3
ทำการบีบอัดข้อมูลด้วยคำสั่ง tar ร่วมกับ bzip2 พร้อมกันในครั้งเดียว ผลลัพธ์ที่ได้คือ ไฟล์ allfile.tar.bz2 เหมือนด้านบน

ลองทดสอบ :

```
# bzip2 -cd foo.tar.bz2 | tar xf -
# tar cjvf filename.tar.bz2 foldername/*
# tar tjvf filename.tar.bz2
# tar  xjvf filename.tar.bz2
```

ดูคำสั่งอื่นประกอบ : *tar, zip*

5.3.8 กลุ่มคำสั่งเกี่ยวกับ File Comparison

คำสั่ง diff

File Comparison : **diff**	
คำสั่ง	**diff** เป็นคำสั่งที่ใช้สำหรับเปรียบเทียบความแตกต่างระหว่าง 2 ไฟล์
Syntax	***diff [options] [diroptions] file1 file2***
Example	**# diff file1 file2** **# diff dir1 dir2**

options:-

-a, --text เปรียบเทียบข้อมูลที่เป็นชนิด text ไฟล์

-b, --ignore-space-change ไม่พิจารณาข้อความที่เป็น spacing

-B, --ignore-blank-lines ไม่พิจารณาข้อความที่มีข้อความว่าง blank

-i, --ignore-case ไม่พิจารณาตัวอักษรตัวเล็กตัวใหญ่(ตัวเล็กและใหญ่จะเหมือนกัน)

-I regexp, --ignore-matching-lines=regexp ไม่สนใจข้อมูลในไฟล์ที่ match กับข้อความที่กำหนดใน regexp

-q, --brief แสดงผลลัพธ์เมื่อไฟล์ทั้ง 2 แตกต่างกัน แบบสรุป

-r, --recursive เปรียบเทียบความแตกต่างในไดเรคทรอรีย่อยๆ ด้วย

-w, --ignore-all-space เปรียบเทียบความแตกต่างโดยไม่สนใจข้อความว่างทั้งหมดในไฟล์

-y, --side-by-side แสดงผลออกเป็นแบบ 2 คอลัมภ์

ตัวอย่างการใช้งาน :

```
file1      file2
aaa        aaa
bbb        bbb
ccc        ccc
ddd        eee

# diff file1 file2
4c4
< ddd
---
> eee
```

เปรียบเทียบความแตกต่างระหว่าง 2 ไฟล์คือ file1 และ file2 จากผลลัพธ์ ไฟล์ทั้ง 2 จะมีความแตกต่างกัน คือ ข้อความ ddd ในไฟล์ file1 และ eee ในไฟล์ file2 สัญลักษณ์ < หมายถึงข้อความที่แตกต่างใน file2 และ > หมายถึงข้อความที่แตกต่างใน file1 และสัญลักษณ์ --- หมายถึงการแบ่งแยกว่าเป็นคนละไฟล์กัน ส่วน 4c4 หมายถึงเปรียบเทียบในบรรทัดที่ 4 ของไฟล์ file1 กับบรรทัดที่ 4 ของไฟล์ file2

```
# diff d1 LAB
Only in LAB: 1
```

Only in LAB: 11
Only in LAB: 2
Only in LAB: 22
Only in LAB: a
Only in LAB: a.txt
Only in LAB: a.txt.gz

เปรียบเทียบความแตกต่างระหว่างไดเรคทรอรี ในกรณีนี้ผลที่ได้จากการเปรียบเทียบจะบอกว่ามีไฟล์อะไรบ้างในไดเรคทรอรีที่แตกต่างกัน โดยไม่ได้พิจารณาในเนื้อหาของไฟล์ภายในไดเรคทรอรี

diff -w file1 file2

เปรียบเทียบความแตกต่างระหว่าง file1 และ file2 โดยไม่สนใจข้อความว่างที่อยู่ในไฟล์ทั้งหมด

diff -by file1 file2

```
aaa         aaa
bbb         bbb
ccc         cc
ddd       | eee
```

เปรียบเทียบความแตกต่างระหว่าง file1 และ file2 โดยไม่สนใจข้อความว่างที่อยู่ในไฟล์ทั้งหมด และแสดงผลแบบ 2 คอลัมภ์

diff -iy file1 file2

เปรียบเทียบความแตกต่างระหว่าง file1 และ file2 โดยไม่สนใจข้อความว่างที่อยู่ในไฟล์ทั้งหมด และไม่สนใจอักษรตัวเล็กตัวใหญ่

ลองทดสอบ :

diff originalfile updatedfile > patchfile.patch

ดูคำสั่งอื่นประกอบ : *bdiff, cmp, comm, dircmp, ed, pr, ls, sdiff*

คำสั่ง comm

File Comparison : **comm**	
คำสั่ง	**comm** เป็นคำสั่งที่ใช้สำหรับเปรียบเทียบไฟล์ที่ได้รับการจัดเรียงลำดับมาแล้ว
Syntax	**comm *[options] file1 file2***
Example	**# comm myfile1.txt myfile2.txt**

ในกรณีที่ต้องการเปรียบเทียบความแตกต่าง ระหว่างไฟล์สองไฟล์ ซึ่งเนื้อหาภายในมีการจัดเรียงลำดับข้อความ หรือตัวเลข ไว้เรียบร้อยแล้ว ระบบปฏิบัติการ unix ได้เตรียมคำสั่ง comm ไว้สำหรับช่วยเปรียบเทียบไฟล์ดังกล่าว ทั้งนี้จำเป็นต้องเข้าใจก่อนว่า ไฟล์ที่จะนำมาเปรียบเทียบกันนั้น ต้องได้รับการจัดเรียงลำดับเนื้อหาภายในไว้ก่อนแล้ว มิฉะนั้นคำสั่ง comm อาจจะทำงานไม่ถูกต้อง ตามที่ต้องการ ผลลัพท์ที่ได้จากการใช้คำสั่งดังกล่าว แบ่งออกเป็น 3 คอลัมภ์ คอลัมภ์ที่หนึ่ง

จะเป็นการแสดง บรรทัดข้อความที่พบเฉพาะไฟล์ file1 คอลัมภ์ที่ สอง แสดงบรรทัดข้อความเฉพาะ ที่พบไฟล์ file2 ส่วนคอลัมภ์ที่ สาม แสดงบรรทัดข้อความที่พบ ทั้ง ในไฟล์ file1 และ file2

options:-

- 1 comm จะไม่แสดงข้อความในคอลัมภ์ที่หนึ่ง (ไม่แสดงข้อความที่พบในไฟล์ file1)
- 2 comm จะไม่แสดงข้อความในคอลัมภ์ที่สอง (ไม่แสดงข้อความที่พบในไฟล์ file2)
- 3 comm จะไม่แสดงข้อความในคอลัมภ์ที่สาม (ไม่แสดงข้อความที่พบในไฟล์ file3)

ตัวอย่างการใช้งาน :

```
File1          File2
aaa            aaa
bbb            bbb
ccc            ccc
ddd            eee
# comm File1 File2
          aaa
          bbb
          ccc
ddd

     eee
```

จากตัวอย่าง File1 และ File2 ข้างต้น เมื่อทำการเปรียบเทียบโดยใช้คำสั่ง comm ผลปรากฏว่า โปรแกรมจะตรวจสอบแบบบรรทัดต่อบรรทัด ในกรณีนี้ aaa, bbb และ ccc ทั้ง 2 ไฟล์เหมือนกัน comm จะแสดงผลในคอลัมภ์ที่ 3 ส่วน ddd(ใน File1 ในคอลัมภ์ที่ 1) และ eee(ใน File2 ในคอลัมภ์ที่ 2) จะมองว่าไม่เหมือนกันจึงแสดงแยกกัน

```
# comm -12 File1 File2
aaa
bbb
ccc
```

ไม่แสดงคอลัมภ์ที่ 1 และ 2 แต่จะแสดงเฉพาะคอลัมภ์ที่ 3 หมายถึงข้อความที่เหมือนกันในตำแหน่งที่ตรงกันด้วย

```
File11         File12
aaa            ccc
bbb            ddd
ccc            bbb
ddd            aaa
# sort File11 > FileSort11; sort File12 > FileSort12
# comm FireSort11 FireSort12
aaa
bbb
ccc
ddd
```

ถ้าต้องการให้การใช้ comm มีประสิทธิภาพจำเป็นต้องมีการ sort ข้อมูลให้เรียบร้อย

เสียก่อนจึงจะทำให้ผลลัพธ์ที่ได้ถูกต้อง

ลองทดสอบ :

```
# comm -12 siskel_top10 ebert_top10
```

ดูคำสั่งอื่นประกอบ : *cmp, diff, sort, uniq*

คำสั่ง cmp

File Comparison : **cmp**	
คำสั่ง	**cmp** เป็นคำสั่งที่ใช้สำหรับเปรียบเทียบไฟล์แบบไบต์ต่อไบต์
Syntax	**cmp [*options*] *file1 file2* [*skip1* [*skip2*]]**
Example	**# cmp file1.txt file2.txt**

options:-

-b --print-bytes แสดงผลความแตกต่างในรูปแบบไบต์

-i num, --ignore-initial=num ไม่ทำการเปรียบเทียบข้อความเท่ากับจำนวน num ไบต์

-l, --verbose แสดงผลลัพธ์ความแตกต่างที่เกิดจากเปรียบเทียบทั้งหมด

-n LIMIT --bytes=LIMIT กำหนดขนาดสูงสุดที่ใช้ในการเปรียบเทียบ

-s --quiet --silent ไม่ต้องแสดงผลใดๆ แสดงเฉพาะรหัสคำสั่งผลของการทำงาน(exit code, 0=เหมือนกัน, 1= แตกต่างกัน, -1 เกิดความผิดพลาด)

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

```
file1          file2
somsri         somsak
manee          manee
sunee          mana
# cmp file1 file2
file1 file2 differ: byte 5, line 1
```

เปรียบเทียบความแตกต่างระหว่างไฟล์ file1 และ file2 ผลลัพธ์จากตัวอย่างแสดงว่า file1 และ file2 มีความแตกต่างกันอยู่ 5 ไบต์ ในบรรทัดที่ 1

```
# cmp -b file1.php file2.php
file1 file2 differ: byte 5, line 1 is 162 r 141 a
```

เปรียบเทียบความแตกต่างระหว่างไฟล์ file1 และ file2 ผลลัพธ์แสดงในรูปแบบไบต์

```
# cmp -l file1 file2
5 162 141
 6 151 153
14 163 155
15 165 141
17 145 141
18 145  12
cmp: EOF on f2
```

ลองทดสอบ :

```
# cmp -c file1.php file2.php
# cmp -s file1 file2 && echo 'no changes'
# cmp -s file1 file1 && echo 'no changes'
```

ดูคำสั่งอื่นประกอบ : *comm, diff*

คำสั่ง md5sum

File Comparison : **md5sum**	
คำสั่ง	**md5sum** เป็นคำสั่งที่ใช้สำหรับตรวจสอบความผิดปกติของไฟล์ที่อาจจะถูกแก้ไขหรือเปลี่ยนแปลงภายหลัง (โดยคำนวณหาค่า md5 ของไฟล์)
Syntax	**md5sum [*option*] [*file*]**
Example	**# md5sum -c file.md5**

options:-

-b, --binary อ่านข้อมูลเข้ามาเปรียบเทียบในโหมดของไบนารี

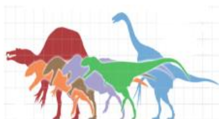
-c, --check อ่านผลรวมของ md5 จากไฟล์พร้อมกับทำการตรวจสอบ

-t, --text อ่านข้อมูลเข้ามาเปรียบเทียบในโหมดของ text (เป็นค่า default)

--status ไม่ต้องแสดงผลใดๆ ส่งเฉพาะรหัสคำสั่งผลการทำงาน(0=, 1= แตกต่างกัน, -1 เกิดความผิดพลาด)

-w, --warn แสดงข้อความผิดพลาดในแต่ละบรรทัดที่เจอความผิดปกติของไฟล์

--help คำสั่งช่วยเหลือ



ข้อควรระวัง: การตรวจสอบค่า md5 นั้นจะต้องทำการตรวจสอบกับไฟล์ที่มี md5 ติดมากับไฟล์เหล่านั้นด้วย ไม่เช่นนั้นจะเกิดข้อผิดพลาดได้ โดยปกติจะทำการตรวจสอบกับไฟล์ประเภท .iso

ตัวอย่างการใช้งาน :

```
# md5sum file1
fe153f61fbc96775b0bbc5815ad9c231  file1
```

ทำการสร้างรหัส md5 ด้วยคำสั่ง md5sum เมื่อส่งไฟล์ให้กับผู้ใช้อื่นๆ จะต้องแนบรหัส md5 ดังกล่าวไปด้วย ปลายทางจะทำการตรวจสอบความถูกต้องของไฟล์ด้วย md5sum กับออพชัน –c

```
# md5sum file1 > file1.md5
```

ทำการสร้างรหัส md5 ของไฟล์ file1 ด้วยคำสั่ง md5sum และส่งไปเก็บไว้ยังไฟล์ชื่อว่า file1.md5 เมื่อส่งไปให้ผู้รับปลายทาง จะต้องส่งไฟล์ไป 2 ไฟล์คือ file1 และ file1.md5

```
# md5sum -c file1.md5
file1: OK
```

ผู้รับปลายทางเมื่อรับไฟล์ file1 และ file1.md5 มาแล้วจะทำการตรวจสอบความถูกต้องของไฟล์ ด้วยคำสั่ง md5sum ออพชัน –c เมื่อไฟล์ดังกล่าวไม่มีข้อผิดพลาดก็จะแสดงข้อความว่า OK

md5sum ubuntu-8.10-desktop-i386.iso > ubuntu-8.10-desktop-i386.md5
24ea1163ea6c9f5dae77de8c49ee7c03 ubuntu-8.10-desktop-i386.iso
สร้างไฟล์ md5 ของไฟล์ระบบปฏิบัติการ ubuntu ชื่อว่า ubuntu-8.10-desktop-i386.iso

md5sum -c ubuntu-8.10-desktop-i386.md5
ubuntu-8.10-desktop-i386.iso: OK
ทำการตรวจสอบความถูกต้องของไฟล์ ubuntu-8.10-desktop-i386.iso

ลองทดสอบ :

md5sum -c CentOS-4.0-i386-bin1of4.md5

5.3.9 กลุ่มคำสั่งเกี่ยวกับ Users and Groups

คำสั่ง useradd

Users and Groups : **useradd**	
คำสั่ง	**useradd** เป็นคำสั่งที่ใช้สำหรับสร้างผู้ใช้รายใหม่และปรับปรุงข้อมูลผู้ใช้
Syntax	**useradd [*options*] [*user*]**
Example	**# useradd newuser** **# useradd -d /home/tom -c "Phd. tom " tom**

options :

-c *comment* ใช้อธิบายความหมายเพิ่มเติม

-d *dir* ระบุถึงตำแหน่งที่อยู่ของผู้ใช้งานที่จะเพิ่ม ถ้าไม่ระบุค่าปริยายจะอยู่ภายใต้ home

-D, --defaults แสดงค่าหรือบันทึกข้อมูลที่ระบบกำหนดไว้เป็นค่า default

-e *date* กำหนดวันหมดอายุของผู้ใช้งานแต่ละราย มีรูปแบบเป็น *MM/DD/YYYY*

-f, --inactive INACTIVE กำหนดรหัสผ่านหลังจากที่รายชื่อผู้ใช้หมดอายุไปแล้ว

-g, --gid GROUP กำหนดรายชื่อกลุ่มสมาชิก

-G, --groups GROUPS กำหนดรายชื่อกลุ่มสมาชิกได้หลายๆ กลุ่ม ใช้ , ในการเพิ่มแบบหลายกลุ่มพร้อมๆ กัน

-h, --help แสดงข้อความช่วยเหลือ

-m สร้าง home ไดเรอทรอรีสำหรับผู้ใช้ใหม่

-M ไม่สร้าง home ไดเรอทรอรีสำหรับผู้ใช้ใหม่

-r สร้างรายชื่อแบบ system account

-o, --non-unique สร้างผู้ใช้ที่มี UID ซ้ำกันได้

-p, --password PASSWORD กำหนดรหัสผ่านให้กับผู้ใช้

-s, --shell SHELL กำหนดเชลล์ให้กับผู้ใช้

-u, --uid UID กำหนดหมายเลข UID ให้ผู้ใช้

-Z, --selinux-user SEUSER กำหนดให้ผู้ใช้ที่มีสิทธิ์ใช้งาน SELinux

ตัวอย่างการใช้งาน :

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

แสดงข้อมูล default ของผู้ใช้งานรายใหม่ทุกคน (กรณีไม่มีการระบุ options)

```
# useradd newperson
newperson:x:1000:1000::/home/newperson:/bin/bash
```

สร้างบัญชีผู้ใช้ใหม่ชื่อว่า newperson คุณสมบัติต่างๆ จะเป็นค่า default เช่น home ไดเรคทรออยู่ที่ /home/newperson, เชลล์คือ bash, UID และ GID ระบบจะสร้างให้อัตโนมัติ สามารถดูข้อมูลผู้ใช้ได้จากไฟล์ /etc/passwd เมื่อสร้างผู้ใช้ใหม่แล้วจะต้องทำการเปลี่ยนรหัสผ่านใหม่เสมอ ด้วยคำสั่ง passwd ตามด้วยชื่อผู้ใช้

```
# useradd -c "Joe Smith" joe
```

สร้างบัญชีผู้ใช้ใหม่ชื่อว่า joe โดยเพิ่มข้อมูลเพิ่มเติม(comment) คือ ชื่อนาม-สกุล

```
# useradd -d /home/james jim
```

สร้างบัญชีผู้ใช้ใหม่ชื่อว่า jim โดยระบุ Home ไดเรคทรอรีชื่อว่า james

```
# useradd -g admin somsak
```

สร้างบัญชีผู้ใช้ใหม่ชื่อว่า somsak โดยเป็นสมาชิกในกลุ่ม admin

```
# useradd -g users -d /home/somchai -c "Mr. somchai boonmee" somchai
```

สร้างบัญชีผู้ใช้ใหม่ชื่อว่า somchai เป็นสมาชิกของกลุ่ม users มี home ไดเรคทรอรีคือ /home/somchai และมีข้อมูลอธิบายเพิ่มเติมคือชื่อ Mr. comchai boonmee

```
# ueradd  Dang -u 1001 -g users -c "Dang Doodee" -d /home/dang
```

สร้างบัญชีผู้ใช้ใหม่ชื่อว่า Dang เป็นสมาชิกกลุ่ม users มีหมายเลข UID คือ 1001, home ไดเรคทรอรีคือ /home/dang มีชื่อว่า Dang Doodee

```
# useradd -D -s /bin/csh
```

เป็นการกำหนดให้ทุกๆ user ใช้เชลล์ csh เป็น default shell ทันทีที่เพิ่ม user เข้าสู่ระบบ

```
# useradd -p abc123 -c user01 -d /home/user01 -m -s /bin/bash -u
501 user01
```

สร้างผู้ใช้ชื่อ user01, รหัสผ่าน abc123, home ไดเรคทรอรีคือ /hoem/user01, เชลล์คือ bash, user ID หมายเลข 501

ลองทดสอบ :

```
# useradd -g root -d /home/user1
```

ดูคำสั่งอื่นประกอบ : *groupadd, passwd, userdel, usermod*

คำสั่ง userdel

Users and Groups : **userdel**	
คำสั่ง	**userdel** เป็นคำสั่งที่ใช้สำหรับลบรายชื่อผู้ใช้พร้อมทั้งคุณสมบัติต่างๆ ออกจากระบบ
Syntax	userdel [*option*] *user*
Example	# userdel newuser # userdel -r newuser

options :

-f, --force ลบรายชื่อผู้ใช้ (บังคับให้ลบผู้ใช้ออกจากระบบให้ได้)

-h, --help ช่วยอธิบายความหมายของคำสั่ง userdel

-r, --remove ลบรายชื่อ พร้อมกับข้อมูลอื่นๆ เช่น home ไดเรคทรอรี, ไฟล์ที่เก็บอีเมล์

EXIT VALUES เมื่อทำการลบผู้ใช้จะมีรหัสแสดงผลลัพธ์ดังนี้

0 – ทำการลบสำเร็จ

1 - ไม่สามารถทำการแก้ไขรหัสผ่านได้

2 - คำสั่งไม่ถูกต้อง

6 - ผู้ใช้ที่ต้องการลบไม่มีในระบบ

8 - ผู้ใช้ที่ลบกำลัง login ใช้งานในระบบอยู่

10 - ไม่สามารถปรับปรุงกลุ่มของผู้ใช้ได้

12 - ไม่สามารถลบ Home ไดเรคทรอรีได้

ตัวอย่างการใช้งาน :

```
# userdel -r newperson
```

ลบข้อมูลรายชื่อผู้ใช้ชื่อ newperson คำสั่งนี้จะไม่ลบ home directory และอีเมล์ออกให้

```
# userdel -r somsri
```

ลบข้อมูลรายชื่อผู้ใช้ชื่อ somsri พร้อมกับลบ home directory และข้อมูลที่เกี่ยวข้องทั้งหมดด้วย

```
# userdel -rf manager
```

ลบข้อมูลรายชื่อผู้ใช้ชื่อ somsri พร้อมกับลบ home directory และข้อมูลที่เกี่ยวข้องทั้งหมดด้วย ถ้ามีข้อผิดพลาดอย่างไร คำสั่งจะทำการลบผู้ใช้ออกจากระบบให้ได้

ดูคำสั่งอื่นประกอบ : *useradd, usermod*

คำสั่ง groupadd

Users and Groups : **groupadd**	
คำสั่ง	**groupadd** เป็นคำสั่งที่ใช้สำหรับสร้างชื่อกลุ่มผู้ใช้ใหม่
Syntax	**groupadd [*options*] *group***
Example	**# groupadd newgroup**

options :

-f, --force บังคับให้มีการสร้างกลุ่มใหม่ให้สำเร็จ ถ้ามีกลุ่มเดิมอยู่จะสร้างทับผู้ใช้เดิม

-r สร้างรายชื่อกลุ่มแบบ system account

-g, --gid GID กำหนดชื่อกลุ่มที่จะสร้างใหม่

-h, --help คำสั่งช่วยอธิบาย groupadd

-K, --key KEY=VALUE เป็น option ที่ให้ทำการเขียนข้อมูลทับ (VALUE) ในไฟล์ชื่อว่า /etc/login.defs ซึ่งไฟล์ดังกล่าวทำหน้าที่เก็บคุณสมบัติของการสร้างผู้ใช้ไว้(default)

-o, --non-unique กำหนดให้ชื่อกลุ่มผู้ใช้สามารถซ้ำกันได้

ตัวอย่างการใช้งาน :

```
# groupadd newgroup
root:x:0:root
bin:x:1:root,bin,daemon
test:x:501:
test1:x:502:
newgroup:x:503:
```

สร้างรายชื่อกลุ่มใหม่ชื่อว่า newgroup ในระบบ เมื่อต้องการดูข้อมูลกลุ่มที่สร้างขึ้นใหม่ สามารถดูได้จากไฟล์ /etc/group

```
# groupadd newgroup
```

สร้างรายชื่อกลุ่มใหม่ชื่อว่า newgroup

```
# groupadd -g 451 newgroup
newgroup:x:451:
```

สร้างรายชื่อกลุ่มใหม่ชื่อว่า newgroup โดยมีหมายเลข GID = 451(ดูข้อมูลผู้ใช้จากไฟล์ /etc/group)

```
# groupadd -o -g 451 work1
newgroup:x:451:
work1:x:451:
```

สร้างรายชื่อกลุ่มใหม่ชื่อว่า work โดยมีหมายเลข GID = 451 ซ้ำกับกลุ่ม newgroup

ลองทดสอบ :

```
# groupadd -g 451 newgroup
# groupadd -g 451 work2
```

ดูคำสั่งอื่นประกอบ : *groupdel, groupmod, useradd*

คำสั่ง groupdel

Users and Groups : **groupdel**	
คำสั่ง	**groupdel** เป็นคำสั่งที่ใช้สำหรับลบรายชื่อกลุ่มออกจากระบบ
Syntax	groupdel *group*
Example	# groupdel newgroup

ตัวอย่างการใช้งาน :

groupdel newgroup
ลบรายชื่อกลุ่ม newgroup ออกจากระบบ

ลองทดสอบ :

```
# groupdel newgroup1 newgroup2
```

ดูคำสั่งอื่นประกอบ : *groupadd, groupmod*

คำสั่ง groupmod

Users and Groups : **groupmod**	
คำสั่ง	**groupmod** เป็นคำสั่งที่ใช้สำหรับแก้ไขหรือปรับปรุงข้อมูลต่างๆ ของกลุ่มผู้ใช้งาน
Syntax	groupmod [*options*] *group*
Example	# groupmod -g 500 regroup

options:-

-g *gid* เปลี่ยนหมายเลข GID ใหม่

-n *name* เปลี่ยนชื่อกลุ่มเป็นชื่อใหม่

-o เขียนข้อมูลทับ ในกลุ่มที่ยอมให้มีข้อมูลซ้ำได้

ตัวอย่างการใช้งาน :

groupmod -n bettergroup newgroup
เปลี่ยนชื่อกลุ่มจากเดิมคือ newgroup ไปเป็น bettergroup

groupmod -g 455 newgroup
เปลี่ยนหมายเลข GID ใหม่ สมมุติว่าเดิม GID ของ newgroup เป็น 451 เมื่อทำคำสั่งดังกล่าว GID จะเปลี่ยนเป็น 455

groupmod -o -g 455 work1
เปลี่ยน GID ของ work1 จากเดิมมี GID=451 ต้องการเปลี่ยนไปเป็น GID=455 ซึ่งจะซ้ำกับ newgroup ถ้าใช้คำสั่งที่ไม่มี –o จะไม่สามารถเปลี่ยน GID ได้ เนื่องจากกลุ่มซ้ำกัน เมื่อใช้ –o จะสามารถเปลี่ยนได้

ลองทดสอบ :

```
# groupmod -g 500 aaa
# groupmod -g 500 bbb
# groupmod -o -g 500 bbb
```

ดูคำสั่งอื่นประกอบ : *groupadd, groupmod*

คำสั่ง grpck

Users and Groups : **grpck**	
คำสั่ง	**grpck** เป็นคำสั่งที่ใช้ตรวจสอบความถูกต้องความซ้ำซ้อนของข้อมูลกลุ่ม ในไฟล์ /etc/group, /etc/gshadow เมื่อพบข้อผิดพลาดจะแสดงข้อความเตือน ถ้ากดปุ่ม yes grpck จะทำการลบข้อมูลที่ผิดพลาดที่เจอนั้นให้ หรืออาจจะแก้ไขผ่านคำสั่ง groupmod แทนก็ได้
Syntax	**grpck [*option*] [*files*]**
Example	**# grpck /etc/group**

options :-

-r กำหนดให้ตรวจสอบไฟล์ แต่ไม่มีการแก้ไข (read-only)

ตัวอย่างการใช้งาน :

grpck /etc/group
ตรวจสอบความถูกต้องของข้อมูลในไฟล์ group
grpck /etc/gshadow
ตรวจสอบความถูกต้องของข้อมูลในไฟล์ gshadow

ดูคำสั่งอื่นประกอบ : *groupadd, groupmod*

คำสั่ง newgrp

Users and Groups : **newgrp**	
คำสั่ง	**newgrp** เป็นคำสั่งที่ใช้เปลี่ยนกลุ่มใหม่ ในขณะใช้งาน
Syntax	**newgrp [*group*]**
Example	**# newgrp newuser**

ตัวอย่างการใช้งาน :

newgrp users
เดิมอยู่กลุ่ม test เมื่อออกคำสั่งดังกล่าวจะเป็นสมาชิกในกลุ่ม users แทน

ดูคำสั่งอื่นประกอบ : *login, set, sh*

คำสั่ง passwd

Users and Groups : **passwd**	
คำสั่ง	**passwd** เป็นคำสั่งที่ใช้สำหรับกำหนดรหัสผ่าน รหัสผ่านบนลินุกซ์นั้น เดิมเก็บไว้ในไฟล์ชื่อว่า /etc/passwd แต่เนื่องจากมีปัญหาด้านความปลอดภัยจึงได้ทำการย้ายรหัสผ่านดังกล่าวมาไว้ในไฟล์ใหม่ชื่อว่า /etc/shadow แล้วมีการเข้ารหัสไว้

Syntax	passwd [*options*] [*user*]
Example	# passwd # passwd user1

options:-

-d, --delete ลบรหัสผ่าน

-f, --force ต้องกำหนดรหัสผ่านให้สำเร็จ

-?, --help คำสั่งช่วยเหลือ

-i *days*, --inactive=*days* กำหนดจำนวนวันที่รหัสผ่านหมดอายุแล้วจะส่งผลให้ ผู้ใช้นั้นไม่สามารถใช้งานได้ (disable user)

-k, --keep-tokens เก็บ token ที่ใช้ในการยืนยันตัวตนไว้ ในกรณีที่ผู้ใช้มีสิทธิ์แบบ non-expired

-l, --lock ทำการล็อกผู้ใช้งานไม่ให้ใช้ระบบไว้ชั่วคราว (root only)

-n *days*, --minimum=*days* กำหนดจำนวนวันที่รหัสผ่านนั้นสามารถใช้งานระบบได้

-S, --status พิมพ์สถิติการใช้งานของผู้ใช้

--stdin อ่านรหัสผ่านใหม่จาก standard input (root เท่านั้น)

-x, --maximum=DAYS กำหนดจำนวนวันสูงสุดของรหัสผ่านที่ใช้งานได้

-w, --warning=DAYS กำหนดจำนวนที่แจ้งเตือนก่อนรหัสผ่านหมดอายุ

--usage แสดงการใช้งานแบบย่อ

-?, --help คำสั่งช่วยการใช้งาน

ข้อมูลในไฟล์ /etc/passwd มีดังนี้

test1:x:502:502:test user:/home/test1:/bin/bash

1 2 3 4 5 6 7

รูปที่ 5-13 แสดงข้อมูลไฟล์ /etc/passwd

ข้อมูลในไฟล์ดังกล่าวจะถูกแยกเป็นส่วนๆ ซึ่งถูกคั่นด้วยเครื่องหมาย : ไว้ดังนี้

1. username ชื่อผู้ใช้
2. password รหัสผ่านของผู้ใช้ จะเข้ารหัสไว้ และเก็บไว้ในไฟล์ชื่อ /etc/shadow

   ข้อมูลในไฟล์ /etc/shadow ที่เข้ารหัสไว้ ไม่สามารถตีความหมายได้

   test1:!!:14563:0:99999:7:::
3. user id (UID) ผู้ใช้แต่ละคนจะต้องมีหมายเลข UID เสมอ หมายเลข UID =0 จะเป็น root, UID 1-99 ถูกใช้สำหรับผู้ใช้ที่รันโปรแกรมของระบบ เช่น ntp, sshd เป็นต้น UID 100-999 ถูกใช้สำหรับทำงานในระดับ system สำหรับผู้ใช้ทั่วไปจะมี UID เริ่มตั้งแต่ 500 เป็นต้นไป
4. group id (GID) เป็นชื่อของกลุ่มสมาชิก (เก็บอยู่ใน /etc/group)

5. user id info เป็น commnent เพื่อขยายหรืออธิบายข้อมูลของ UID นั้นๆ

6. home directory เป็นพื้นที่หรือ ไดเรคทรอรีบ้านของผู้ใช้งานแต่ละคนไว้ (ส่วนใหญ่จะอยู่ที่ /home แต่สามารถเปลี่ยนได้ โดยใช้ option –d ขณะทำการเพิ่มผู้ใช้)

7. command shell เป็นเชลล์ที่ผู้ใช้ใช้ในการสั่งงานระบบปฏิบัติการ

หมายเหตุ: สามารถดูรายละเอียดได้จากการใช้คำสั่ง useradd

ตัวอย่างการใช้งาน :

```
# passwd
Current Password:       ← ใส่รหัสผ่านปัจจุบัน
New Password:           ← ป้อนรหัสผ่านใหม่
Confirm New Password:   ← ยืนยันรหัสผ่านใหม่อีกครั้ง
เปลี่ยนรหัสผ่านใหม่
# passwd newperson
สั่งให้ทำการเปลี่ยนรหัสผ่านของผู้ใช้ชื่อ newperson ใหม่
# passwd -d sleepy
ลบรหัสผ่านของผู้ใช้งานชื่อ sleepy ออก
# passwd -x 60 test1
Adjusting aging data for user test1.
passwd: Success
กำหนดเวลาการใช้งานของผู้ใช้ test1 สูงสุดไม่เกิน 60 วัน
# passwd -n 10 test1
Adjusting aging data for user test1.
passwd: Success
กำหนดเวลาการใช้งานของผู้ใช้ test1 อย่างต่ำไม่น้อยกว่า 10 วัน
# passwd -w 3 test1
Adjusting aging data for user test1.
passwd: Success
กำหนดเวลาการแจ้งเตือนก่อนรหัสผ่านหมดอายุ อย่างน้อย 3 วัน
```

ลองทดสอบ :

```
# passwd -x 60 test1
# passwd -n 60 test1
```

ดูคำสั่งอื่นประกอบ : *chfn, finger, login, nispasswd, nistbladm, useradd, vipw, yppasswd*

คำสั่ง pwck

Users and Groups : **pwck**	
คำสั่ง	**pwck** เป็นคำสั่งที่ใช้ตรวจสอบความถูกต้องของ syntax และ format ของไฟล์ /etc/passwd
Syntax	**pwck [*options*] [*files*]**

Example	# pwck /etc/passwd

options:-

-q แสดงรายงานเฉพาะข้อมูลที่มีความผิดปกติชนิดร้ายแรง

-r ตรวจสอบไฟล์แบบอ่านอย่างเดียวโดยไม่มีการแก้ไข เมื่อเจอความผิดพลาด

-s ไม่ต้องตรวจสอบความถูกต้อง แต่ให้เรียงลำดับการแสดงตามลำดับ UID

-files เลือกตรวจสอบได้ว่าจะตรวจสอบไฟล์อะไร ระหว่าง /etc/passwd หรือ /etc/shado

ตัวอย่างการใช้งาน :

```
# pwck /etc/passwd
user adm: directory /var/adm does not exist
user news: directory /etc/news does not exist
user uucp: directory /var/spool/uucp does not exist
user gopher: directory /var/gopher does not exist
user ftp: directory /var/ftp does not exist
```

ตรวจสอบ syntax ของไฟล์ที่เก็บรายชื่อผู้ใช้ จากตัวอย่าง user adm มีชื่อในระบบแต่ไม่มีไดเรคทรอรี ให้ผู้ใช้ดังกล่าวทำงาน

```
# pwck /etc/shadow
```

ตรวจสอบ syntax ของไฟล์ที่เก็บรหัสผ่านที่เข้ารหัสไว้

ลองทดสอบ :

```
# pwck -q /etc/passwd
# pwck -r /etc/passwd
# pwck -rq /etc/passwd
# pwck myfile.txt
```

ดูคำสั่งอื่นประกอบ : *cat, compress, pcat, tar, unpack, zcat*

5.3.10 กลุ่มคำสั่งเกี่ยวกับการเปลี่ยนทิศทาง(Redirection) และการเชื่อมต่อกันระหว่าง input กับ output(Pipe)

คำสั่งลินุกซ์หลายคำสั่ง จะให้ผลลัพธ์ของการใช้คำสั่งออกทางหน้าจอคอมพิวเตอร์ (เรียกกันว่า standard output จะใช้หมายเลข 1) การรับข้อมูลก็จะรับมาจากการพิมพ์ผ่านแป้นคีย์บอร์ด (เรียกว่า standard input ใช้หมายเลข 0) และในกรณีที่มีเกิดการผิดพลาดในการประมวลผลคอมพิวเตอร์จะแสดงข้อผิดพลาดออกมาทาง standard error ซึ่งปกติจะเป็นอันเดียวกับ standard output ซึ่งก็คือจอภาพ (เรียกว่า standard error นี้จะแทนด้วยเลข 2) ในบางกรณีผู้ดูแลระบบต้องการให้ผลลัพธ์ของการใช้คำสั่ง ไปเก็บไว้ที่ไฟล์ หรือส่งออกทางอีเมลล์ หรือถ้าเป็นส่วนของการรับข้อมูล ก็สามารถเปลี่ยนจากการรับจากแป้นคีย์บอร์ด มาเป็นรับจากไฟล์บ้าง ซึ่งกระบวนการเหล่านี้เรียกว่า Redirection

คำสั่ง >, >>

Standard input/Output(Redirection) : **>, >>**

คำสั่ง	**>** เป็นคำสั่งที่ใช้สำหรับเปลี่ยนเส้นทางในการแสดงผล (Output) สัญลักษณ์ที่ใช้แทนการส่งออกข้อมูลคือ > (แบบเขียนทับข้อมูลเดิม) **>>** เป็นคำสั่งที่ใช้สำหรับเปลี่ยนเส้นทางในการแสดงผล (Output) สัญลักษณ์ที่ใช้แทนการส่งออกข้อมูลคือ >> (แบบเขียนข้อมูลต่อจากเดิม)
Syntax	**command [file] > file**
Example	**# cat > myfile** **# ls -al > ls.txt**

ตัวอย่างการใช้งาน :

cat > fruit
Apple
Banana
Mango
กด (Ctrl + d) เพื่อบันทึกและออกจากการเขียนไฟล์

คำสั่ง cat โดยปกติแล้ว จะแสดงข้อมูลในไฟล์ แล้วแสดงออกมาทางจอภาพ แต่ในบางครั้ง เราต้องการให้คำสั่ง cat เขียนข้อมูลลงบนไฟล์ ตัวอย่างเช่น สร้างไฟล์ชื่อว่า fruit ไว้เก็บชื่อผลไม้ต่างๆ เมื่อใส่รายชื่อเสร็จแล้ว ให้กดปุ่ม ^d (Ctrl + d) เพื่อบันทึกและออกจากการเขียนไฟล์

cat >> fruit
Orange
Pineapple
กด (Ctrl + d) เพื่อบันทึกและออกจากการเขียนไฟล์

ถ้าหากต้องการเพิ่มข้อมูลลงในไฟล์เดิม โดยที่ข้อมูลไม่หายไป (Append) ให้เปลี่ยนเส้นทางจาก > ไปเป็น >> เพื่อเป็นการระบุว่า จะทำการเขียนไฟล์ต่อจากเดิม

ผลลัพธ์ที่ได้คือ

Apple
Banana
Mango
Orange
Pineapple

คำสั่ง <

Standard input/Output(Redirection) : **<**	
คำสั่ง	**<** เป็นคำสั่งที่ใช้สำหรับการเปลี่ยนเส้นทางที่ฝั่งรับข้อมูล (Input)
Syntax	**command *[file]* < *file***
Example	**# sort < myfile**

ตัวอย่างการใช้งาน :

สมมุติว่าไฟล์ข้อมูล number มีข้อมูลคือ

```
6
4
2
7
8
# sort < number
2
4
6
7
8
```

การเรียงข้อมูลในไฟล์ใหม่ ด้วยใช้คำสั่ง sort โดยเรานำไฟล์ที่ยังไม่มีการจัดเรียง มาเป็นอินพุต (input) ของคำสั่ง sort

คำสั่ง pipe(|)

Standard input/Output(Redirection) : **Pipe(|)**	
คำสั่ง	| (ไปป์) เป็นคำสั่งที่ใช้สำหรับการเชื่อมต่อกันระหว่าง input กับ output โดย output ของคำสั่งหนึ่ง จะเป็น input ของอีกคำสั่งหนึ่ง จะใช้สัญลักษณ์เป็น | (Vertical Bar)
Syntax	**command1 | command2**
Example	**# ps -ef | grep sshd** **# grep ftp services | less**

ตัวอย่างการใช้งาน :

grep ftp services | less

ตัวอย่างนี้เป็นการหาข้อความ ftp ที่อยู่ในไฟล์ services แต่เนื่องจากมีข้อมูลอยู่เป็นจำนวนมาก ทำให้ไม่สามารถดูได้ทั้งหมดในครั้งเดียว จึงอาศัยคำสั่ง less เพื่อจัดเรียงข้อมูลให้สามารถอ่านได้ง่ายขึ้น

cat /etc/passwd | grep ftp

แสดงข้อมูลที่อยู่ในไฟล์ passwd จากนั้นนำ output ที่ได้จาก cat ไปเป็น input ของคำสั่ง grep ซึ่งเป็นคำสั่งที่เลือกบรรทัดที่มีคำว่า ftp อยู่ด้วยมาแสดงผล

5.3.11 กลุ่มคำสั่งเกี่ยวกับ Disks and Filesystems

คำสั่ง df

Disks and Filesystems : **df**	
คำสั่ง	**df** เป็นคำสั่งที่ใช้สำหรับแสดงรายงานการใช้เนื้อที่บนฮาร์ดิสก์ ซึ่งประกอบไปด้วย

	ปริมาณที่ใช้ ปริมาณที่เหลือ และรายการพาร์ติชัน
Syntax	**df [*options*] [*name*]**
Example	**# df** **# df -h**

option :

-a, --all แสดงรายงานการใช้ดิสก์รวมถึงไฟล์ประเภท dummy filesystem ด้วย

-B, --block-size=SIZE แสดงขนาดของดิสก์ตามที่ระบุใน SIZE

-h, --human-readable รายงานการใช้ดิสก์ให้ผู้ใช้สามารถอ่านและทำความเข้าใจได้ง่ายขึ้น เช่น แสดงหน่วยเป็นกิโลไบต์(K), เม็กกะไบต์(M), กิกกะไบต์(G)

-H, --si เหมือนการใช้ออพชัน –h แต่แตกต่างที่ –H จะคำนวณเลขกำลังเป็นฐานสิบแทนฐานสอง เช่น –h จะใช้ 2^{10} = 1024 แต่ –H จะใช้ 10^{3} = 1000 แทน

-i, --inodes แสดงพื้นที่ที่ใช้งาน พื้นที่ว่าง และ inodes

-k แสดงหน่วยเป็นกิโลไบต์

-l, --local แสดงเฉพาะไฟล์ที่เป็น local file system

-m แสดงหน่วยเป็นเม็กกะไบต์

-t, --type=TYPE แสดงเฉพาะระบบไฟล์ที่ถูกกำหนดใน TYPE เช่น df –t ext3

-T, --print-type สั่งพิมพ์ประเภทของไฟล์

-x type, --exclude-type=type แสดงระบบไฟล์ที่ไม่ได้กำหนดใน type เช่น df –x ext3 จะแสดงระบบไฟล์ที่ไม่ใช้ ext3

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

```
# df
Filesystem          1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      6983168   2571300   4051420  39% /
/dev/sda1              101086     11885     83982  13% /boot
tmpfs                  257744         0    257744   0% /dev/shm
```

แสดงรายงานการใช้พื้นที่ดิสก์ พื้นที่เหลือ และพาร์ติชัน รูปแบบการรายงานจะมีขนาดเป็น 1 กิโลไบต์ ต่อ block ถ้าต้องการรู้ขนาดจริงต้องนำจำนวน block คูณด้วย 1024

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      6.7G  2.5G  3.9G  39% /
/dev/sda1              99M   12M   83M  13% /boot
tmpfs                 252M     0  252M   0% /dev/shm
```

แสดงรายงานการใช้พื้นที่ดิสก์ พื้นที่เหลือ และพาร์ติชัน ในรูปแบบที่ทำความเข้าใจได้ง่าย คือ แสดงหน่วยเป็น กิโลไบต์ หรือ เม็กกะไบต์แทนการรายงานแบบ blocks

```
# df -Tha
Filesystem    Type    Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
              ext3    6.7G  2.5G  3.9G  39% /
proc          proc       0     0     0   -  /proc
sysfs         sysfs      0     0     0   -  /sys
devpts        devpts     0     0     0   -  /dev/pts
/dev/sda1     ext3     99M   12M   83M  13% /boot
tmpfs         tmpfs   252M     0  252M   0% /dev/shm
none   binfmt_misc       0     0     0   -  /proc/sys/fs/binfmt_misc
sunrpc rpc_pipefs        0     0     0   -  /var/lib/nfs/rpc_pipefs
```

จากตัวอย่างเป็นการแสดงการใช้ดิสก์ โดยแสดงไฟล์ทั้งหมดที่ใช้ในระบบ (-a) ง่ายต่อการทำความเข้าใจ(-h) และพิมพ์ชื่อชนิดของไฟล์ เช่น ext3, sysfs(-T)

```
#  df -h /
```

รายงานการใช้ดิสก์ โดยการระบุพาร์ติชันที่ต้องการ(ในที่นี้แสดงผลพาร์ติชัน /)

ลองทดสอบ :

```
# df -h /dev/sda1
# df -h /dev/hda1
# df -i
# df -H
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *du, find, ls*

คำสั่ง du

Disks and Filesystems : **du**	
คำสั่ง	**du** เป็นคำสั่งที่ใช้สำหรับแสดงเนื้อที่การใช้งาน ของแต่ละไดเรคทรอรีแบบละเอียด
Syntax	**du [*options*] [*files*]**
Example	**# du myfile.txt** **# du -s myfile.txt**

options:-

--a, --all แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลทั้งหมด

-b, --bytes แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลมีหน่วยเป็นไบต์

-c, --total แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลแบบสรุปผลรวม

-h, --human-readable แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลที่เหมาะสมและอ่านเข้าใจง่าย

-k แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลมีหน่วยเป็นกิโลไบต์ (1 กิโลไบต์มีขนาดเท่ากับ 1,024 ไบต์)

-m แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลมีหน่วยเป็นเม็กกะไบต์ (1 เม็กกะไบต์มีขนาดเท่ากับ 10,240 ไบต์)

-s, --summarize แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลโดยสรุป

--help คำสั่งช่วยเหลือ

ตัวอย่างการใช้งาน :

du /etc/passwd
```
4       /etc/passwd
```
แสดงเนื้อที่การใช้งานของไฟล์ /etc/passwd

du -s *.txt
```
8   file1.txt
8   file2.txt
10  file3.txt
2   file4.txt
```
แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลทั้งหมดที่มีส่วนขยายเป็น .txt ในไดเรคทรอรีปัจจุบันโดยแสดงแยกกันในแต่ละไฟล์

du -ch *.txt
```
4.0K   a.txt
4.0K   b.txt
4.0K   file1.txt
4.0K   log.txt
4.0K   myfile.txt
13M    s.txt
20K    text.txt
4.0K   vim.txt
4.0K   vi.txt
4.0K   wc.txt
13M    total
```
แสดงเนื้อที่การใช้งานของไฟล์ข้อมูลแต่ละไฟล์ให้ผู้ใช้งานอ่านและทำความเข้าใจได้โดยง่าย(แสดงขนาดของข้อมูลเป็น กิโลไบต์/เม็กกะไบต์ ตามความเหมาะสม) และในบรรทัดสุดท้ายจะสรุปรวมขนาดของข้อมูลด้วย

du /root/LAB /root/LAB1
```
4       /root/LAB/mydir
14552   /root/LAB
4       /root/LAB1/mydir
12520   /root/LAB1
```
แสดงเนื้อที่การใช้งานของไดเรคทรอรี /root/LAB และ /root/LAB1 พร้อมกัน

du -h /root/LAB
```
4.0K   /root/LAB/mydir
```

```
15M     /root/LAB
```

แสดงเนื้อที่การใช้งานของไดเรคทรอรี /root/LAB โดยออพชัน –h จะทำให้ผู้ใช้สามารถทำความเข้าใจกับขนาดของพื้นที่การใช้งานได้ง่ายขึ้น ถ้าไดเรคทรอรีมีไดเรคทรอรีย่อยๆ คำสั่งดังกล่าวจะแสดงขนาดของแต่ละไดเรคทรอรีย่อยให้ด้วย

```
# du -sh /root/LAB
15M     /root/LAB
```

แสดงเนื้อที่การใช้งานของไดเรคทรอรี /root/LAB แบบสรุปรวบยอด โดยไม่แสดงขนาดของไดเรคทรอรีย่อยๆ

ลองทดสอบ :

```
# du *
# du -hs *
# du -h s*
# du -hc s*
# du -hc *.gif
# du --max-depth=1
# du | sort -n
# du -ah | grep M
# du -sk .[A-z]* *| sort -n
# du -sk * | sort -nr | head -3
# du * | sort -n
# info du
# du -sh ~
```

ดูคำสั่งอื่นประกอบ : *df, ls*

คำสั่ง mount, umount

Disks and Filesystems : **mount, umount**	
คำสั่ง	**mount** เป็นคำสั่งที่ใช้สำหรับเชื่อมต่ออุปกรณ์จัดเก็บข้อมูล หรือพาร์ติชัน เข้าสู่ระบบ ตัวอย่างเช่ CD-ROM, Diskette, Flash drive, เป็นต้น
Syntax	**mount [*options*] [[*device*] *directory*]**
Example	**# mount -F dos /dev/rdsk/fds0d2.3.5 /floppy** **# umount /dev/fd0**

option :

-a เชื่อมต่ออุปกรณ์หรือพาร์ติชันทั้งหมดที่ระบุไว้ในไฟล์ /etc/fstab แต่ถ้าไม่ต้องการเชื่อมต่อทั้งหมด ต้องใช้ร่วมกับ –t ในการระบุชนิดของอุปกรณ์ที่ต้องการเชื่อมต่อด้วย

--bind olddirectory newdirectory ผูกโครงสร้างของไดเรคทรอรีที่ถูก mount ไว้แล้วเข้ากับโครงสร้างของไดเรคทรอรีใหม่

-f ทดสอบการเชื่อมต่อเพื่อตรวจสอบความผิดพลาด แต่ยังไม่มีกระบวนเชื่อมต่อจริงๆ

-F ใช้งานคู่กับ –a โดยออพชันดังกล่าวจะสร้างโพรเซสใหม่สำหรับการเชื่อมต่อตามจำนวนของอุปกรณ์ที่ต้องการเชื่อม (fork a new process) ทำให้ mount ได้เร็วขึ้น

-l ขณะทำการเชื่อมต่ออุปกรณ์จะแสดงชื่อระบบไฟล์ที่เชื่อมต่อ

-L label เชื่อมต่ออุปกรณ์โดยการระบุชื่อใน Label

--move olddirectory newdirectory ทำการย้ายตำแหน่งอุปกรณ์ที่เชื่อมต่อไว้แล้วในระบบไปยังตำแหน่งใหม่

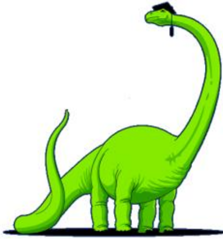
-n ไม่ต้องบันทึกรายการเชื่อมต่อลงยังไฟล์ /etc/mtab

-o option ระบุเงื่อนไขหรือรูปแบบการเชื่อมต่อในแต่ละประเภทของอุปกรณ์ โดยมีออพชันให้เลือกดังต่อไปนี้

options	**ความหมาย**
async	อ่านและเขียนข้อมูลโหมด asynchronously
atime	ปรับปรุงตารางเวลาที่มีการเข้าถึงอุปกรณ์
auto	อนุญาติให้ mount ด้วยออพชัน –a ได้
defaults	เหมือนกับ auto ค่าที่ mount โดย default คือ async, auto, dev, exec, nouser, rw, suid
dev	เชื่อมต่อกับระบบโครงสร้างไฟล์ที่มีอยู่ในระบบ
dirsync	
exec	ประมวลผลแบบไบนารี
_netdev	โครงสร้างของไฟล์ระบบที่ใช้สำหรับเชื่อมต่อกับเครือข่าย
noatime	ไม่ปรับปรุงค่าของ inodes ที่ใช้งาน
noauto	ไม่อนุญาติให้ mount ด้วยออพชัน –a
noexec	ไม่อนุญาติให้ประมวลผลแบบไบนารี
remount	เมื่ออุปกรณ์เชื่อมต่อแล้วจะทำการเชื่อมต่อซ้ำอีกครั้ง
ro	อนุญาตให้อ่านอุปกรณ์เชื่อมต่อได้อย่างเดียว
rw	อนุญาตให้อ่านและเขียนอุปกรณ์เชื่อมต่อได้
sync	อ่านและเขียนอุปกรณ์เชื่อมต่อแบบเข้าจังหวะกัน (synchronously)

-t type ระบุชนิดของไฟล์ระบบที่ต้องการเชื่อมต่อ ซึ่งมีให้เลือกใช้งานได้หลายประเภท เช่น adfs, affs, autofs, coda, cramfs, devpts, efs, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, nfs4, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xfs, and xiafs ค่า default คือ iso9660 ออพชันแบบ auto

--help คำสั่งช่วยเหลือ



NOTE: คำสั่ง mount และ unount จะมีความเกี่ยวข้องกับไฟล์ระบบ 3 ไฟล์ คือ

/etc/fstab :แสดงรายการไฟล์ระบบทั้งหมดที่ถูก mount ไว้

/etc/mtab :แสดงรายการไฟล์ระบบทั้งหมดที่ถูก mount ในปัจจุบัน พร้อมกับออพชันที่เลือก

/etc/filesystems: เป็นไฟล์ระบบที่เก็บข้อมูลระบบไฟล์ที่ลินุกซ์รู้จักและสนับสนุน

/proc/partitions:เป็นไฟล์ระบบที่เก็บ ชื่อ(label) และขนาดของแต่ละพาร์ติชัน

/proc/mounts: เป็นไฟล์ระบบที่เก็บข้อมูลที่ได้ mount ไว้แล้ว

/sbin/fdisk -l : แสดงพาร์ติชัน จากฮาร์ดิสก์ทุกตัวที่เชื่อมต่อในระบบ

การ mount นั้นสามารถกระทำได้กับพาร์ติชันหรืออุปกรณ์เชื่อมต่อได้หลายแบบ เช่น mount ไดเรคทรอรีเดิมที่มีอยู่เข้ากับไดเรคทรอรีใหม่, mount floppy disk, ฮาร์ดดิสก์, flash drive, CD/DVD, แทบ เป็นต้น

ตัวอย่างการใช้งาน :

mount

```
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
```

แสดงข้อมูลที่ได้ทำการ mount ไว้ทั้งหมด

ตัวอย่างการ mount ไดเรคทรอรีที่มีอยู่แล้ว

mkdir /ext

สร้างไดเรคทรอรีใหม่ เพื่อใช้สำหรับการ mount ชื่อว่า /ext

mount --bind /var/www /ext

mount

```
/var/www on /ext type none (rw,bind)
```

ทำการ mount ไดเรคทรอรี /var/www ซึ่งมีอยู่แล้วในระบบ เข้ากับระบบไฟล์ที่สร้างขึ้นใหม่(/ext) ในไดเรคทรอรี /ext เมื่อผ่านการ mount แล้วจะมีข้อมูลเหมือนกับไดเรคทรอรี /var/www ทุกประการ เมื่อต้องการดูรายละเอียดที่เกิดจากการ mount ให้ใช้คำสั่ง mount

umount /ext

ยกเลิกการ mount ไดเรคทรอรี /ext

วิธีใช้แผ่น Floppy Disk

mkdir /mnt/floppy

สร้างไดเรคทรอรีใหม่ เพื่อใช้สำหรับการ mount floppy ชื่อว่า /mnt

/floppy

mount /dev/fd0 /mnt/floppy

ทำการ mount อุปกรณ์ floppy disk เข้ากับโครงสร้างไฟล์ชื่อว่า /mnt

/floppy

mount –t ext3 /dev/fd0 /mnt/floppy

ทำการ mount อุปกรณ์ floppy disk เข้ากับโครงสร้างไฟล์ชื่อว่า

/mnt/floppy โดยระบุชนิดของโครงสร้างไฟล์ที่เชื่อมต่อคือ ext3

umount /dev/fd0

ยกเลิกการ mount floppy disk

การ Mount DOS diskettes

mount -t msdos /dev/fd0 /mnt/floppy

ทำการ mount อุปกรณ์ floppy disk แบบ dos เข้ากับโครงสร้างไฟล์ชื่อว่า

/mnt/floppy

mount -t vfat dev/fd0 /mnt/floppy

การ mount flash drive

mkdir /mnt/flash

สร้างไดเรคทรอรีใหม่ เพื่อใช้สำหรับการ mount flash ชื่อว่า /flash

mount /dev/sda1 /mnt/flash

ทำการ mount อุปกรณ์ flash drive(/dev/sda1) เข้ากับโครงสร้างไฟล์ชื่อ

ว่า /mnt/flash

umount /dev/sda1

ยกเลิกการ mount flash

การ mount CD-Rom

mount /dev/cdrom

ทำการ mount อุปกรณ์ cd-rom เข้ากับโครงสร้างไฟล์ชื่อว่า /mnt(เป็น

โครงสร้างไฟล์ที่ถูกสร้างในตอนติดตั้งระบบ เป็นค่า default)

mkdir /mnt/cdrom

สร้างไดเรคทรอรีใหม่ เพื่อใช้สำหรับการ mount cd-rom ชื่อว่า /mnt

/cdrom

mount /dev/cdrom /mnt/cdrom

ทำการ mount อุปกรณ์ cd-rom drive(/dev/hdc) เข้ากับโครงสร้างไฟล์ชื่อว่า /cdrom

mount -t iso9660 /dev/hdc /mnt/cdrom

ทำการ mount อุปกรณ์ cd-rom drive อีกแบบหนึ่ง

umount /dev/cdrom

ยกเลิกการ mount cd-rom

eject

สั่งดีด CD-ROM ออกจากตัวเครื่อง

การ mount พาร์ติชัน เช่น Windows

mkdir /mnt/hd0

สร้างไดเรคทรอรีใหม่ เพื่อใช้สำหรับการ mount ฮาร์ดดิสก์ ชื่อว่า /mnt /hd0

fdisk -l

แสดงรายการพาร์ติชันที่มีอยู่ในระบบ เพื่อใช้สำหรับทำการ mount เข้ากับ /hd0

mount /dev/hdb3 /mnt/hd0

ทำการ mount อุปกรณ์ฮาร์ดดิกส์ /dev/hdb3 เข้ากับ /mnt/hd0

umount /hd0

ยกเลิกการ mount /hd0

การ mount รูปแบบอื่น ๆ

$ mount -o remount,rw /dev/hda2

ทำการ remount โครงสร้างไฟล์เดิมอีกครั้ง ให้สามารถอ่านและเขียนได้



NOTE: โดยปกติเมื่อใช้คำสั่ง mount กับโครงสร้างไฟล์แบบใดแบบหนึ่งแล้วเมื่อ restart เครื่องใหม่ โครงสร้างที่ mount ไว้ดังกล่าวจะหายไป ต้องมีการ mount ใหม่ แต่ถ้าต้องการให้ mount ทุกครั้งเมื่อมีการเริ่มต้นทำงานใหม่ ให้กำหนดรายละเอียดการ mount ไว้ในไฟล์ระบบชื่อว่า /etc/fstab หรือ /ect/mtab ดังนี้

/dev/sda1 /mnt/flash auto noauto,user 0 0 จากตัวอย่างทำการกำหนดว่าเมื่อเริ่มต้นการทำงานของเครื่องทุกครั้งให้ทำการ mount โครงสร้างไฟล์ /dev/sda1 เข้ากับ /mnt/flash ชนิด auto เสมอ สำหรับการดูรายละเอียดโครงสร้างไฟล์ที่เชื่อมต่อไว้แล้ว เราสามารถใช้คำสั่ง mount –t filetype ได้ เช่น monut –t ext3, mount –t ext2, mount –vfat เป็นต้น

ลองทดสอบ :

```
# mount --move olddir newdir
# /dev/cdrom /cd iso9660 ro,user,noauto,unhide
# mount -a -t nomsdos,ext
```

คำสั่ง fsck [34]

Disks and Filesystems : **fsck**	
คำสั่ง	fsck เป็นคำสั่งที่ใช้สำหรับตรวจสอบและซ่อมแซมความผิดปรกติของเนื้อที่ดิสก์
Syntax	**fsck [*options*] [*filesystem*]**
Example	# fsck # fsck -t ext2 /dev/fd0

option :

-p ซ่อมแซมดิสก์ให้แบบอัตโนมัติ

-t fstype ระบุชนิดของระบบไฟล์ที่ต้องการตรวจสอบ เช่น ext3

-y ตรวจสอบและซ่อมแซมดิกส์ในลักษณะ ถาม-ตอบ

ตัวอย่างการใช้งาน :

fsck
สั่งให้ทำการตรวจสอบความผิดปกติของไฟล์ทั้งหมดที่ประกาศไว้ใน /etc/filesystems โปรแกรมจะถามผู้ใช้ขณะทำการตรวจสอบโครงสร้างไฟล์แต่ละประเภท เพื่อยืนยันว่าผู้ใช้ต้องการตรวจสอบระบบไฟล์เหล่านั้นหรือไม่

fsck -p
สั่งซ่อมแซมพื้นที่ดิสก์ที่ผิดปกติในเบื้องต้นแบบอัตโนมัติ

fsck /dev/hd1
สั่งให้ทำการตรวจสอบความผิดปกติของข้อมูลใน /dev/hd1(ฮาร์ดดิกส์ชนิด IDE)

fsck /dev/sdb1
สั่งให้ทำการตรวจสอบความผิดปกติของข้อมูลใน /dev/sdb1 (ฮาร์ดดิกส์ชนิด SCSI หรือ SATA)

fsck -t ext3 /dev/sda3
สั่งให้ทำการตรวจสอบความผิดปกติของข้อมูลใน /dev/sdb3 โดยการระบุเงื่อนไขเป็นชนิดของไฟล์แบบ ext3

fsck -y /dev/sda3
สั่งให้ทำการตรวจสอบและแก้ไขความผิดปกติของข้อมูลใน /dev/sdb3

ในรูปแบบถาม-ตอบ ถ้าตรวจสอบแล้วพบความผิดปกติจะให้แก้ไขด้วยหรือไม่ ถ้ากด y คือ ให้แก้ไข ถ้ากด n ให้ข้ามความผิดปกตินั้นไป

ลองทดสอบ :

```
# df -h /dev/sda1
# df -h /dev/hda1
# df -i
# df -H
```

คำสั่งหรือไฟล์ที่เกี่ยวข้องอื่นๆ :

/usr/sbin/fsck ตำแหน่งที่อยู่ของโปรแกรม fsck

/etc/filesystems ไฟล์ระบบที่เก็บข้อมูลชนิดของระบบไฟล์ที่ระบบปฏิบัติการรู้จัก

/etc/rc เพิ่มคำสั่งที่ต้องการให้ระบบประมวลผลเมื่อเริ่มต้นการทำงานของระบบ

คำสั่งอื่นที่ทำงานคล้ายกัน : *mkfs*

5.3.12 กลุ่มคำสั่งเกี่ยวกับ Backups and Remote Storage

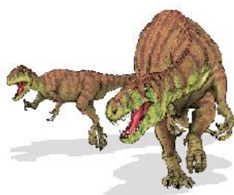
คำสั่ง mt

Backups and Remote Storage : **mt**	
คำสั่ง	**mt** เป็นคำสั่งที่ใช้สำหรับควบคุมเทปสำหรับการสำรอง(backup) หรือกู้คืนข้อมูล (restore)
Syntax	**mt [*option*] *operation* [*count*] [*arguments*]**
Example	**# mt -f /dev/st0 status**

option :

-f device, -t device กำหนดอุปกรณ์ที่ใช้สำรอง/กู้คืนข้อมูล

-t fstype ระบุชนิดของระบบไฟล์ที่ต้องการตรวจสอบ เช่น ext3



NOTE: อุปกรณ์จัดเก็บข้อมูลประเภทเทป เมื่อเชื่อมต่อบนลินุกซ์จะมีชื่ออุปกรณ์เป็น /dev/stx โดย x หมายถึงจำนวนของเทปที่เชื่อมต่อกับระบบ เช่น /dev/st0 หมายถึง เทปตัวที่ 1 เป็นต้น

ตัวอย่างการใช้งาน :

mt -f /dev/st0 rewind

สั่งให้เทปตัวที่ 1 เล่นเทปแบบย้อนกลับ

mt -f /dev/st0 tell

เป็นคำสั่งที่ใช้สำหรับตรวจสอบว่าจะใช้ block ข้อมูลใดในเทป ที่สามารถใช้งานได้บ้าง

tar -tzf /dev/st0

สั่งให้แสดงไฟล์ข้อมูลบนเทป
mt -f /dev/st0 status
แสดงสถานะการทำงานของเทป
mt -f /dev/nst0 eod
สั่งให้เทปไปยัง block สุดท้ายของข้อมูลที่อยู่ในเทป
mt -f /dev/nst0 bsfm 1
สั่งให้เทปย้อนกลับไป 1 record
mt -f /dev/nst0 fsf 1
สั่งให้เทปเดินหน้าไป 1 record
mt -f /dev/st0 erase
สั่งให้ทำการลบข้อมูลในเทปทั้งหมด
mt -f /dev/st0 offline
สั่งให้เทปหยุดการทำงาน เพื่อทำการเปลี่ยนเทปใหม่
tar -czf /dev/st0 /www /home
สั่งสำรองข้อมูลของ /www/ และ /hom ลงในเทป ด้วยคำสั่ง tar(ดูรายละเอียดเพิ่มเติมได้จากคำสั่ง tar)
mt -f /dev/st0 rewind; tar -xzf /dev/st0 www
สั่งกู้คืนข้อมูลของ www กลับคืน ด้วยคำสั่ง tar

ลองทดสอบ :

```
# tar -clpMzvf /dev/st0 /home
# tar -xlpMzvf /dev/st0 /home
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *tar, tcopy*

คำสั่ง dump

Backups and Remote Storage : **dump**	
คำสั่ง	**dump** เป็นคำสั่งที่ใช้สำหรับการสำรองจากฮาร์ดิกส์(backup)
Syntax	**dump [*options*] *files***
Example	**# dump -0f databackup /home/user1/data** **# dump -1f databackup /home/user1/data**

option :

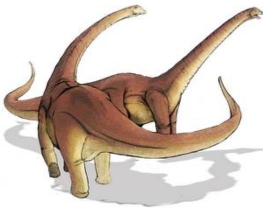
-[level] กำหนดรูปแบบการสำรองข้อมูล ซึ่งจะเป็นค่าตัวเลขจำนวนเต็ม คือ

0 สำรองข้อมูลแบบ full backup สำรองข้อมูลทั้งหมด

1 สำรองข้อมูลแบบ incremental backup สำรองข้อมูลเฉพาะที่มีการเปลี่ยนแปลง

-f กำหนดไดเรคทรอรีหรือไฟล์ข้อมูลที่ต้องการสำรอง

-u สั่งให้ทำการบันทึกข้อมูลการสำรองลงไฟล์ระบบชื่อ /etc/dumpdats



NOTE: เมื่อผู้ใช้ต้องการสำรองข้อมูล ในครั้งแรกต้องทำการสำรองแบบ full backup(สำรองข้อมูลทั้งหมด) เสียก่อนเนื่องจากยังไม่เคยมีการสำรองข้อมูลมาก่อน หลังจากนั้นจะใช้การสำรองข้อมูลแบบ incremental backup แทน(สำรองข้อมูลเฉพาะส่วนที่มีการเปลี่ยนแปลง)

ตัวอย่างการใช้งาน :

dump -0uf databackup /home/user1/data สั่งสำรองข้อมูลด้วยคำสั่ง dump จากตัวอย่างจะทำการสำรองข้อมูลในไดเรคทรอรี /home/user1/data ไฟล์ที่ถูกสำรองแล้วชื่อว่า databackup โดยออพชัน -0 หมายถึงสำรองข้อมูลแบบ full backup(เป็นการสำรองข้อมูลเป็นครั้งแรก จำเป็นต้องสำรองข้อมูลทุกอย่างที่มี) พร้อมบันทึกการสำรองลงไฟล์ /etc/dumpdats **# dump -0f databackup /home/user1/data** ไม่บันทึกข้อมูลการสำรองลงไฟล์ /etc/dumpdats **# dump -1uf databackup /home/user1/data** สั่งสำรองข้อมูลด้วยคำสั่ง dump โดยออพชัน -1 หมายถึงสำรองข้อมูลแบบ incremental backup(สำรองข้อมูลในส่วนที่มีการเปลี่ยนแปลงเท่านั้น) **# dump -1f databackup /home/user1/data** ไม่บันทึกข้อมูลการสำรองลงไฟล์ /etc/dumpdats

คำสั่งอื่นที่ทำงานคล้ายกัน : *tar, mt, restore*

คำสั่ง restore

Backups and Remote Storage : **restore**	
คำสั่ง	**restore** เป็นคำสั่งที่ใช้สำหรับกู้คืนข้อมูล(restore)ที่สำรองไว้ด้วยคำสั่ง dump กลับคืนมา
Syntax	**restore [*options*] *files***
Example	**# restore -if databackup**

option :

-C เปรียบเทียบไฟล์ที่สำรองไว้(dump-file) กับไฟล์ต้นฉบับ

-f กำหนดไดเรคทรอรีหรือไฟล์ข้อมูลที่ต้องการกู้คืนข้อมูล

-i กู้คืนข้อมูลกลับในโหมด interactive

-v แสดงข้อมูลการกู้คืนทั้งหมด

เมื่อใช้ออพชัน –i จะทำการกู้คืนข้อมูลในโหมด interactive ซึ่งผู้ใช้สามารถใช้คำสั่งอื่นๆ ได้เพิ่มเติม ดังนี้

ls แสดงรายการไฟล์ข้อมูลที่สำรองไว้ใน dump-file

add กำหนดไดเรคทรอรีที่ต้องการกู้คืนข้อมูล

cd เปลี่ยนไดเรคทรอรี

pwd แสดงไดเรคทรอรีที่กำลังทำงานอยู่

extract แตกไฟล์ที่สำรองไว้(dump-file) ไปยังไดเรคทรอรีที่ต้องการในดิสก์

quit ออกจากโหมด interactive

ตัวอย่างการใช้งาน :

restore -if databackup
สั่งกู้คืนข้อมูลด้วยคำสั่ง restore จากตัวอย่างจะทำการกู้คืนข้อมูลจากไฟล์ที่สำรองไว้ด้วยคำสั่ง dump คือ databackup โดยออพชัน -if หมายถึงสำรองข้อมูลในโหมด interactive เมื่อเข้าไปสู่โหมดการทำงานดังกล่าวแล้ว(จะมีเครื่องหมาย prompt เป็น restore>) ให้ทำขั้นตอนดังนี้ เพื่อกู้คืนข้อมูลกลับมา

restore> ls
แสดงรายการไฟล์ข้อมูลที่สำรองไว้ใน dump-file

restore> add
กำหนดไดเรคทรอรีที่ต้องการกู้คืนข้อมูล(ในที่นี้คือไดเรคทรอรีที่กำลังทำงานอยู่)

restore> ls
แสดงรายการไฟล์ข้อมูลที่เพิ่มเข้าไปจาก dump-file ในไดเรคทรอรีปัจจุบัน

restore> extract
สั่งให้ทำการแตกไฟล์ที่สำรองไว้ในไดเรคทรอรีปัจจุบัน

restore> quit
ออกจากโหมด interactive

restore -Cf databackup
สั่งให้ทำการเปรียบเทียบไฟล์ที่สำรองไว้ในไดเรคทรอรีปัจจุบัน

คำสั่งอื่นที่ทำงานคล้ายกัน : *dump*

คำสั่ง tar

Backups and Remote Storage : **tar**

คำสั่ง	**tar** เป็นคำสั่งที่ใช้เพื่อการสำรองข้อมูล (backup) และการกู้คืน (restore) การ tar จะเก็บทั้งโครงสร้างไดเรคทรอรี และสิทธิ์ต่างๆ ของไฟล์ (permission) ด้วย (เหมาะสำหรับการเคลื่อนย้าย หรือแจกจ่ายโปรแกรมบนระบบ UNIX) ย่อมาจากคำว่า tape archive
Syntax	**tar [*options*] [*tarfile*] [*other-files*]**
Example	**# tar -cvf home.tar /home/** **# tar -cvf home.tar.gz /home/** **# tar -xvf myfile.tar** **# tar -xvf myfile.tar.gz**

options:-

-c, --create สร้าง backup file หรือไดเรคทรอรี

-d, --diff, --compare เปรียนเทียบไฟล์ที่อยู่ใน tar ไฟล์กับไดเรคทรอรีอื่น และแสดงรายงานความแตกต่าง เช่น ไฟล์ที่หายไป, ขนาดของไฟล์, คุณลักษณะที่แตกต่างกัน

-f *file*, --file=*file* ระบุชื่อไฟล์ที่จะสำรองหรือกู้คืน

-j, --I, --bzip ทำการบีบอัดไฟล์ด้วยวิธีบีบอัดแบบ bzip2 และคลายไฟล์ด้วย bunzip2

-r, --append เพิ่มไฟล์ที่ต้องการสำรองต่อท้าย ไฟล์ที่เคยสำรองก่อนหน้าไว้แล้ว

-t, --list แสดงรายชื่อไฟล์ข้อมูลในไฟล์ที่สำรองไว้

-u, --update เพิ่มไฟล์เข้าไปใหม่ในกรณีที่ไฟล์ยังไม่เคยทำการสำรอง แต่ถ้ามีอยู่แล้วจะ update ไฟล์ใหม่เข้าไป

-x, --extract, --get แตกไฟล์จากที่สำรองไว้ออกมา หรือกู้คืนไฟล์กลับมา(restore)

-A, --catenate, --concatenate เชื่อม tar ไฟล์ 2 ไฟล์เข้าด้วยกัน

-C *directory*, --directory=*directory* ระบุตำแหน่งไดเรคทรอรีที่ต้องการ ก่อนใช้ tar

-v, --verbose แสดงผลการสำรองและกู้คืนข้อมูลทั้งหมด

-z, --gzip, --gunzip, --ungzip ทำการบีบอัดไฟล์ก่อนทำการสำรองไฟล์ หรือทำการแตกไฟล์ที่บีบอัดไว้ก่อนกู้คืนไฟล์

ตัวอย่างการใช้งาน :

tar -cvf file.tar myfile.txt
ทำการสำรองไฟล์ข้อมูล ชื่อว่า myfile.txt ไปเป็นไฟล์สำรองชื่อ file.tar

tar -cvf home.tar home/
ทำการสำรองไดเรคทรอรี ชื่อว่า home ไปเป็นไฟล์สำรองชื่อ home.tar

tar -xvf myfile.tar
ทำการกู้คืนไฟล์ข้อมูลที่สำรองไว้ ชื่อว่า myfile.tar กลับคืนในไดเรคทรอรีปัจจุบัน

tar -xvf home.tar
ทำการกู้คืนไดเรคทรอรีที่สำรองไว้ ชื่อว่า home.tar กลับคืนในไดเรคทรอรีปัจจุบัน

tar -cvzf myfile.tar.gz myfile.txt

ทำการบีบอัดข้อมูลก่อนสำรองไฟล์ข้อมูล ชื่อว่า myfile.txt ไปเป็น myfile.tar.gz

tar -xvzf myfile.tar.gz

ทำการคลายไฟล์บีบอัดข้อมูลก่อนกู้คืนไฟล์ข้อมูล ชื่อว่า myfile.tar.gz ไปเป็น ไฟล์ชื่อ myfile.txt ในไดเรคทรอรีปัจจุบัน

tar -cvzf /backup/myfile.tar.gz /home/myfile.txt

ทำการบีบอัดข้อมูลก่อนสำรองไฟล์ข้อมูล อยู่ใน /home/ myfile.txt ไปเก็บไว้ใน /backup/ ชื่อว่า myfile.tar.gz

tar -xvzf /backup/myfile.tar.gz

ทำการคลายไฟล์ที่บีบอัดข้อมูลก่อนกู้คืนไฟล์ข้อมูล อยู่ใน /backup/myfile.tar.gz ไปเก็บไว้ในตำแหน่งเดิมที่ไฟล์นั้นเคยติดตั้งอยู่คือ (สมมุติว่าอยู่ที่ /home/)

tar -cjvf test.tbz home/

ทำการสำรองไฟล์ข้อมูลร่วมกับการบีบอัดข้อมูลโดยใช้เทคนิคการบีบอัดแบบ bzip2 ของไดเรคทรอรีชื่อว่า home ไปเป็นไฟล์ที่สำรองแบบบีบอัดชื่อว่า test.tbz

tar -xjvf test.tbz home/

คลายไฟล์ที่สำรองร่วมกับการบีบอัดข้อมูลโดยใช้เทคนิคการบีบอัดแบบ bzip2 ของไฟล์ชื่อว่า test.tbz ไปเก็บไว้ยัง ไดเรคทรอรีชื่อว่า home

tar -tf mybackup.tar

แสดงรายชื่อไฟล์ที่ทำการสำรองไว้ ใน mybackup.tar โดยไม่ต้องมีการคลายไฟล์ออกมา

tar cvfM /dev/fd0 panda

ทำการสำรองไฟล์ข้อมูลจากฮาร์ดดิสก์ ชื่อว่า panda ไปเก็บไว้ใน flopy disk

tar cvf /dev/rmt0 panda

ทำการสำรองไฟล์ข้อมูลจากฮาร์ดดิสก์ ชื่อว่า panda ไปเป็นเก็บไว้ในเทป

tar cvzf foo.tgz *.cc *.h

ทำการสำรองไฟล์พร้อมบีบอัดข้อมูลทุกไฟล์ที่มีส่วนขยายเป็น .cc และทุกไฟล์ที่มีส่วนขยายเป็น .h ไปเป็นไฟล์ที่สำรองชื่อ foo.tgz

tar cvzf foo.tgz cps100

ทำการคลายไฟล์ที่สำรองพร้อมบีบอัดไว้ ชื่อว่า foo.tgz ออกเป็นไดเรคทรอรีชื่อ cps100

tar cvf /dev/rmt0 /bin /usr/bin

ทำการสำรองไฟล์ข้อมูลใน /bin และ /user/bin ไปไว้ในเทป

tar -xvf archive.tar -C /tmp

ทำการกู้คืนข้อมูล archive.tar ไปไว้ในไดเรคทรอรี /tmp

tar -cvfj archive.tar.bz2 dir1

ทำการสำรองพร้อมบีบอัดข้อมูลแบบ bz2 ในไดเรคทรอรี dir1 เป็นชื่อ archive.tar.bz2

tar -xvfj archive.tar.bz2

คลายไฟล์บีบอัดข้อมูลแบบ bz2

tar -cvfj archive.tar.bz2 dir1

ทำการสำรองพร้อมบีบอัดข้อมูลแบบ bz2 ในไดเรคทรอรี dir1 เป็นชื่อ archive.tar.bz2
tar -cvfj archive.tar.bz2 dir1
ทำการสำรองพร้อมบีบอัดข้อมูลแบบ bz2 ในไดเรคทรอรี dir1 เป็นชื่อ archive.tar.bz2

ลองทดสอบ :

```
# tar --create --verbose --file=afiles.tar apple angst aspic
# tar -c -v -f afiles.tar apple angst aspic
# tar --create --verbose --file=afiles.tar apple angst aspic
```

ดูคำสั่งอื่นประกอบ : *ar, basename, cd, chown, cpio, csh, dirname, ls, mt, pack, pax, setfacl, umask, zcat*

คำสั่ง rsync

Backups and Remote Storage : **rsync**	
คำสั่ง	**rsync** เป็นคำสั่งที่ใช้สำหรับสำรองข้อมูลระหว่างโฮสต์
Syntax	**rsync [*options*] *sources dest***
Example	**# rsync -v -e ssh /backup/backup01 suchart@10.114.224.55:~**

option :

-a archive mode

--delete ลบไฟล์ที่อยู่ปลายทางที่ไม่มีชื่อเหมือนกับไฟล์ที่ต้องการส่งจากต้นทางทิ้งทั้งหมด

-v แสดงรายละเอียดการสำรองข้อมูล

-e "ssh options" สำรองข้อมูลโดยเรียกใช้งานผ่าน ssh (เป็นการส่งข้อมูลแบบปลอดภัยโดยมีการเข้ารหัสข้อมูล)

-r, --recursive สำรองข้อมูลในไดเรคทรอรีย่อยทั้งหมด

-z บีบอัดข้อมูลก่อนการสำรองข้อมูล



NOTE: ก่อนทำการสำรองข้อมูลระหว่างโฮสต์ เครื่องปลายทางที่จะสำรองจะต้องมีการติดตั้งโปรแกรม sync server เสียก่อน สำหรับการติดตั้ง sync server ทำได้ดังนี้

กรณีลินุกซ์ใช้ Debian หรือ Ubuntu

apt-get install rsync

กรณีลินุกซ์ใช้ CentOS/Fedora Core

yum install rsync

ตัวอย่างการใช้งาน :

```
# rsync -v -e ssh /www/backup.tar.gz
suchart@archive.msu.ac.th:~
suchart@archive.msu.ac.th's password: xxx
backup.tar.gz
```

sent 1095901 bytes received 42 bytes 70706.00 bytes/sec
total size is 1095680 speedup is 1.00
สั่งสำรองข้อมูลระหว่างโฮสต์จากเครื่องผู้ใช้ไปยัง sync server ด้วยคำสั่ง rsync ไฟล์ที่ต้องการสำรองชื่อ backup.tar.gz โดยใช้ออพชัน –v คือ แสดงผลการสำรองข้อมูล –e เรียกใช้โปรแกรม secure shell เพื่อทำการส่งข้อมูลโดยการเข้ารหัส ซึ่งทำให้ข้อมูลมีความปลอดภัยและเป็นที่นิยม ในตัวอย่างจะใช้ผู้ใช้ชื่อ suchart ซึ่งต้องเป็นรายชื่อผู้ใช้และรหัสผ่านที่มีอยู่ในระบบในโฮสต์ปลายทางด้วย เมื่อยืนยันตัวตนถูกต้อง ข้อมูลจะถูกส่งไปเก็บไว้ใน home ไดเรคทรอรีของผู้ใช้ suchart

rsync -v -e ssh suchart@archive.msu.ac.th:~/ backup.tar.gz /tmp

สั่งสำรองข้อมูลจากโฮสต์ sync server กลับมายังเครื่องผู้ใช้

rsync -r -a -v -e "ssh -l suchart" --delete archive.msu.ac.th:/webroot/ /local/webroot

สั่งให้ทำการ syncronize(ตรวจสอบความแตกต่างและปรับปรุงข้อมูลให้เหมือนกัน) ระหว่างเครื่อง sync server กับเครื่องผู้ใช้งาน ในไดเรคทรอรีชื่อว่า webroot

rsync -r -a -v -e "ssh -l suchart" --delete /local/webroot archive.msu.ac.th:/webroot/

สั่งให้ทำการ syncronize(ตรวจสอบความต่างและปรับปรุงข้อมูลให้เหมือนกัน) ระหว่างเครื่องผู้ใช้งาน กับเครื่อง sync server กับ ในไดเรคทรอรีชื่อว่า webroot

ลองทดสอบ :

```
# rsync -r -a -v --delete rsync://rsync.msu.ac.th/cvs /home/cvs
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *rcp*

5.3.13 กลุ่มคำสั่งเกี่ยวกับ Printing

คำสั่ง lpr, lpq, lprm

Printing : **lpr, lpq, lprm**	
คำสั่ง	lpr เป็นคำสั่งที่ใช้สำหรับสั่งพิมพ์ไปยังเครื่องให้บริการพิมพ์(print server)
Syntax	lpr [*options*] [*filename*]
Example	# lpr myfile.txt

option :

-H server[:port] ระบุเครื่องที่ให้บริการการพิมพ์

-C title, -J title, -T title กำหนดชื่อการพิมพ์ในแต่ละครั้ง(job name) เพื่อจะได้ทราบว่างานใดกำลังถูกพิมพ์อยู่

-U username ระบุชื่อผู้สั่งพิมพ์

-P ระบุเครื่องพิมพ์ที่ต้องการ กรณีเครื่องดังกล่าวมีเครื่องพิมพ์มากกว่า 1 เครื่อง

-# num-copies กำหนดจำนวนการพิมพ์

-o option[=value] กำหนดเงื่อนไขการพิมพ์

-p กำหนดรูปแบบในการพิมพ์

-q หยุดการพิมพ์ชั่วคราว

-J กำหนดชื่อ job name

ตัวอย่างการใช้งาน :

cat thesis.txt > /dev/lp

ส่งผลลัพธ์ที่ได้จากคำสั่ง cat คือข้อมูลที่อยู่ในไฟล์ thesis.txt พิมพ์ออกทางเครื่องพิมพ์(ต้องทำการติดตั้งเครื่องพิมพ์ไว้ที่เครื่องด้วย)

lpr thesis.txt

สั่งพิมพ์ข้อมูลในไฟล์ thesis.txt ออกทางเครื่องพิมพ์

lpq

```
lp is ready and printing
Rank   Owner      Job  Files                Total Size
active mwf        31   thesis.txt           682048 bytes
```

แสดงข้อมูลการพิมพ์ในคิว(print queue)

lprm -

ยกเลิกรายการที่กำลังพิมพ์ในคิวทั้งหมด

$ lprm 31

ยกเลิกรายการที่กำลังพิมพ์ในคิว ซึ่งเป็น job ที่ 31 คือ thesis.txt

lpr -J "my hosts file" /etc/hosts

สั่งพิมพ์ข้อมูลในไฟล์ /etc/hosts ออกทางเครื่องพิมพ์ โดยตั้งชื่อ job name คือ “my hosts file”

lpr -P mylaserjet /etc/services

สั่งพิมพ์ข้อมูลในไฟล์ /etc/services ออกทางเครื่องพิมพ์ โดยระบุชื่อเครื่องพิมพ์ด้วยออพชัน –P ชื่อ mylaserjet

lpr -#2 thesis.txt

สั่งพิมพ์ข้อมูลในไฟล์ thesis.txt ออกทางเครื่องพิมพ์ ซ้ำกัน 2 ชุด

คำสั่งอื่นที่ทำงานคล้ายกัน : *hostname, lp, lpc, lpq, lprm, lpstat, mail*

5.3.14 กลุ่มคำสั่งเกี่ยวกับ Processes management

คำสั่ง ps

Processes management : **ps**	
คำสั่ง	**ps** เป็นคำสั่งที่ใช้สำหรับแสดงโพรเซส (Process) หรือโปรแกรมที่กำลังประมวลผลอยู่ในระบบ
Syntax	**ps [*options*]**
Example	**# ps** **# ps -ef**

option :

-a แสดงรายละเอียดของโพรเซสทั้งหมดในเทอร์มินัลที่กำลังทำงานอยู่

-A หรือ e แสดงรายละเอียดของโพรเซสทั้งหมดที่กำลังทำงานอยู่ในระบบ

-d แสดงรายละเอียดของโพรเซสทั้งหมดยกเว้นโพรเซส session leaders

-f แสดงรายละเอียดของโพรเซสเต็มรูปแบบ(full listing)

-j แสดง session ID และโพรเซส group ID

-u แสดงชื่อผู้สั่งงานโพรเซส

-l แสดงรายละเอียดของโพรเซสแบบยาว(long listing)



NOTE: การใช้คำสั่ง ps จะแสดงอักษรย่อ เช่น PID, UID ซึ่งมีรายละเอียดดังนี้

PID(process ID) หมายเลขประจำโปรเซส

TTY(terminal) แสดงหมายเลขประจำเครื่องหรือ เทอร์มินัล ซึ่งเป็นผู้ที่ทำให้เกิดโปรเซสนี้ขึ้น

S(state) ใช้ตัวอักษรซึ่งมีอยู่ทั้งสิ้น 4 ตัว สำหรับชี้แสดงถึงสถานะของโปรเซส ในปัจจุบันที่พบเห็นอยู่บ่อยๆ ก็ได้แก่ S(หยุดการทำงานไปไม่ถึง 20 วินาที) I (ไอ) (หยุดการทำงานไปมากกว่า 20 วินาที) และ R (ยังทำงานอยู่)

TIME แสดงเวลานับตั้งแต่โปรเซสดังกล่าวได้รับการสร้างขึ้น มีหน่วยเป็น นาที : วินาที

COMMAND แสดงชื่อของคำสั่งที่สร้างโปรเซสนี้ขึ้น

F(flags) แสดงรายละเอียดของแฟล็กที่เกี่ยวข้องกับโปรเซสแต่ละชุด

UID(user ID) แสดงหมายเลข user ID ซึ่งเป็นเจ้าของโปรเซสนั้น ๆ

PD(process ID) หมายเลขประจำโปรเซส ของโปรเซสนั้น ๆ

PPID(parent PID) แสดงหมายเลข PID ของโปรเซสที่เป็นผู้ให้กำเนิดโปรเซสนั้น ๆ

C(central processor utilization) ระบบปฏิบัติการยูนิกซ์ จะใช้หมายเลขนี้ในการสร้างหมายกำหนดการสำหรับ ดำเนินการสั่งการโปรเซส

RPI(priorty) แจ้งถึงระดับความสำคัญของโปรเซส ค่ามากจะมีระดับความน้อย

NI(nice) เป็นตัวเลข ที่ถูกสร้างขึ้น โดยคำสั่ง nice ใช้สำหรับการคำนวณ หาระดับความสำคัญของโปรเซส

SZ(size) รายงานถึงขนาดของส่วนประกอบหลักของโปรเซส มีหน่วยเป็นบล็อค

WCHAN(wail channel) รายงานโปรเซสที่กำลังหยุดทำงานชั่วคราว(wait)

ตัวอย่างการใช้งาน :

```
# ps
PID TTY         TIME CMD
 6137 pts/2    00:00:00 bash
18582 pts/2    00:00:00 ps
```
แสดงโพรเซสที่กำลังทำงาน โดยแสดงเฉพาะโพรเซสที่ผู้ใช้กำลังใช้งาน

```
# ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
root      6137  6135  0 03:07 pts/2    00:00:00 -bash
root     18588  6137  0 05:00 pts/2    00:00:00 ps -f
```
แสดงโพรเซสที่กำลังทำงานเต็มรูปแบบ

```
# ps –ef
UID        PID  PPID  C STIME TTY          TIME CMD
root      4660     1  0 02:34 ?        00:00:00 /usr/sbin/sshd
root      4671     1  0 02:34 ?        00:00:00 cupsd
root      4683     1  0 02:34 ?        00:00:00 xinetd -stayalive -pidfile
/var/run/xinetd.pid
root      4703     1  0 02:34 ?        00:00:00 sendmail: accepting connections
smmsp     4711     1  0 02:34 ?        00:00:00 sendmail: Queue runner@01:00:00
for /var/spool/clientmqueue
root      4723     1  0 02:34 ?        00:00:00 gpm -m /dev/input/mice -t exps2
root      4734     1  0 02:34 ?        00:00:00 crond
```
แสดงข้อมูลของโพรเซสที่กำลังทำงานในระบบ โดยละเอียด

```
# ps –ax
  PID TTY      STAT   TIME COMMAND
 4683 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
 4703 ?        Ss     0:00 sendmail: accepting connections
 4711 ?        Ss     0:00 sendmail: Queue runner@01:00:00 for
/var/spool/clientmqueue
 4723 ?        Ss     0:00 gpm -m /dev/input/mice -t exps2
```
แสดงข้อมูลของ process พร้อมชื่อโปรแกรมอย่างละเอียด

```
# ps –aux
  USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
  root         1  0.0  0.1   2080   636 ?        Ss   02:31   0:02 init [5]
  root         2  0.0  0.0      0     0 ?        S<   02:31   0:00 [migration/0]
  root         3  0.0  0.0      0     0 ?        SN   02:31   0:00 [ksoftirqd/0]
  root         4  0.0  0.0      0     0 ?        S<   02:31   0:00 [watchdog/0]
```
แสดงข้อมูลของ process พร้อมชื่อโปรแกรม ชื่อผู้สั่ง ซีพียู

หน่วยความจำ และข้อมูลอื่นๆ อย่างละเอียด

```
# ps -ef | less
```

แสดงข้อมูลของ process อย่างละเอียด และส่งผลลัพธ์ที่ได้ให้กับคำสั่ง less เพื่อให้สามารถแสดงรายงานให้มีความยืดหยุ่นขึ้น

ลองทดสอบ :

```
# ps -ef | grep adam
# ps -l
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *bg, kill, nice, pagesize, pgrep, priocntrl, top, uptime, who*

คำสั่ง w

Processes management : **w**	
คำสั่ง	**w** เป็นคำสั่งที่ใช้สำหรับแสดงรายชื่อยูสเซอร์ที่กำลังใช้งานโปรเซส
Syntax	**w [*options*] [*user*]**
Example	**# w**

option :

-h ไม่แสดงส่วนหัวของคอลัมภ์

-l แสดงรายละเอียดแบบ long listing(default)

-s แสดงรายละเอียดแบบสั้น

ตัวอย่างการใช้งาน :

```
# w
06:21:43 up  3:50,  1 user,  load average: 0.00, 0.01, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/2    192.168.1.2       03:07    0.00s  1.17s  0.06s w
```

แสดงโพรเซสที่กำลังทำงาน ข้อมูลที่แสดงประกอบด้วย USER คือเจ้าของ คือผู้ที่สร้างโพรเซส, TTY คือเทอร์มินัลที่ใช้งานอยู่, FROM ผู้ใช้เชื่อมต่อมาจากระยะไกลคือไอพี 192.168.1.2, เวลาที่ทำการ Login, ปริมาณการใช้งานซีพียู และโพรเซสที่ประมวลผล เป็นต้น

```
# w -h
root     pts/2    192.168.1.2       03:07    0.00s  1.27s  0.12s w -h
```

แสดงโพรเซสที่กำลังทำงาน เหมือนข้างบน แต่จะไม่แสดงชื่อของคอลัมภ์

คำสั่งอื่นที่ทำงานคล้ายกัน : *who*

คำสั่ง uptime

Processes management : **uptime**	
คำสั่ง	**uptime** เป็นคำสั่งที่ใช้สำหรับแสดงปริมาณภาระการทำงานของระบบ
Syntax	**uptime [*option*]**
Example	**# uptime**

option :

-V แสดงเวอร์ชัน

ตัวอย่างการใช้งาน :

```
# uptime
14:49:07 up     199 days,     4:15,    105 users,     load average:
0.16,    0.52,    0.34
```

แสดงปริมาณภาระการทำงานของระบบ โดยแสดงเวลาของระบบ, สถานะการทำงาน, จำนวนวันที่ระบบเริ่มทำงาน, จำนวนผู้ใช้งานในระบบ, ภาระการทำงานของระบบ, ชั่วโมง / นาที ที่ระบบทำงาน

คำสั่งอื่นที่ทำงานคล้ายกัน : *vmstat*

คำสั่ง top

Processes management : **top**	
คำสั่ง	**top** เป็นคำสั่งที่ใช้สำหรับแสดงข้อมูลที่เกี่ยวข้องกับโปรเซสแบบต่อเนื่อง
Syntax	**top [*options*]**
Example	**# top**

option :

-d delay กำหนดช่วงเวลาในการ refreshes ข้อมูลใหม่

-f เพิ่มหรือลดจำนวนคอลัมภ์ที่ต้องการแสดงผล

-h คำสั่งช่วยเหลือการใช้งาน

-p pid ตรวจสอบเฉพาะโพรเซสที่ต้องการเฝ้าระวัง ด้วยการระบุโดยใช้ pid

-u user ตรวจสอบโพรเซสที่สั่งโดยผู้ใช้งานชื่อ user เท่านั้น

การใช้คำสั่ง top นั้นผู้ใช้งานสามารถตอบโต้กับโปรแกรมได้ ซึ่งจะมีโหมดการทำงานดังนี้

คำสั่งที่ใช้สำหรับโต้ตอบกับโปรแกรม top

space, Enter สั่งให้ top ทำการ refreshes ข้อมูลใหม่ทันที

<, > เลื่อนข้อมูลที่กำลังแสดงผลขึ้นและลง เมื่อใช้ > ให้เลื่อนข้อมูลลงทีละหนึ่งบรรทัด และ < ให้เลื่อนขึ้น

A แสดงผลการทำงานของโพรเซสสลับกันระหว่าง 1 หน้าจอภาพกับหลายๆ จอภาพ

B แสดงผลคอลัมภ์เป็นอักษรหนาและทึบ

c แสดงผลโดยดูจากชื่อของคำสั่งหรือคำสั่งที่ใช้งานแบบเต็มสลับกัน

d, s เปลี่ยนเวลาการแสดงผลให้เร็วขึ้นหรือช้าลง โดยการกดปุ่ม d หรือ s ขณะโปรแกรม top ทำงาน จะปรากฏช่องว่างให้ใส่เวลาที่ต้องการ refresh มีหน่วยเป็นวินาที

f เพิ่มหรือลบคอลัมภ์ที่ต้องการแสดงผล เมื่ออยู่ในโปรแกรม top ให้ใช้คำสั่ง f โปรแกรมจะแสดงข้อมูลให้เลือกว่าจะเพิ่มหรือลบคอลัมภ์ใดบ้าง เช่น A: PID คือ Process Id, E: USER คือ User Name โดยปกติคำสั่ง top จะแสดงคุณสมบัติเกือบทั้งหมดของโพรเซสออกมาคือใช้คอลัมภ์ AEHIOQTWKNMbcdfgjplrsuvyzX เมื่อผู้ใช้ออกคำสั่ง f และตามด้วย A จะทำให้คอลัมภ์ของ process id หายไปจากคำสั่ง top เมื่อต้องการให้กลับมาใหม่ให้กด A อีกครั้ง สำหรับคำสั่งอื่นๆ ก็ทำนองเดียวกัน

F, O เลือกคอลัมภ์ที่ต้องการจัดเรียงใหม่

G สร้างกลุ่มของรายการที่ต้องการเฝ้าระวังใหม่ จากเดิมคำสั่ง top จะมีการเฝ้าระวังทุกๆ คอลัมภ์ แต่คำสั่ง G จะอนุญาตให้สร้างกลุ่มใหม่ได้ โดยมีให้เลือก 4 กลุ่มคือ 1=def, 2=job, 3=mem และ 4=Usr

h แสดงคำสั่งที่ใช้งานแบบ real-time

k สั่งให้โพรเซสที่ต้องการหยุดการทำงาน

i สลับการแสดงผลการทำงานระหว่างโหมดของโพรเซสที่ถูกสั่งงานโดยผู้ใช้(idle) กับโหมดที่โพรเซสของระบบทำงานอยู่(zombie)

l แสดงผลระหว่างข้อมูลภาระการทำงานของระบบ(load-average) และข้อมูลการทำงานโดยเริ่มนับจากเวลาตั้งแต่ระบบเริ่มต้นทำงาน(uptime information)

m แสดงผลการใช้หน่วยความจำ โดยเรียงจากมากไปน้อยหรือน้อยไปมาก

N เรียงตามหมายเลข PID

o จัดเรียงตามชื่อคอลัมภ์ที่ต้องการ

P จัดเรียงตามการใช้งาน CPU

q ออกจากโปรแกรม top

w บันทึกการทำงานลง log ไฟล์ โดยปกติจะเก็บไว้ใน ~/.toprc

ความหมายของแต่ละคอลัมภ์จากคำสั่ง top

top แสดงเวลาของระบบตั้งแต่เริ่มต้นทำงาน, จำนวนของผู้ใช้งานในระบบ และปริมาณการใช้งานของระบบโดยรวมที่จะเกิดขึ้นอีก 1, 5 และอีก 15 นาทีข้างหน้า เช่น

top - 19:34:04 up 2:31(เวลาของระบบตั้งแต่เริ่มต้นทำงาน), 1 user(จำนวนของผู้ใช้งานในระบบ), load average: 0.02(งานใน 1 นาที), 0.03(งานใน 5 นาที), 0.00(งานใน 15 นาที)

Tasks แสดงจำนวนของโพรเซสทั้งหมดที่กำลังทำงานอยู่ในระบบ หลังจากมีการ refresh ครั้งล่าสุด โดยแสดงโพรเซสที่กำลังทำงาน, หยุดทำงานชั่วคราว(sleeping), หยุดทำงานแล้ว(stoped) และโพรเซสของระบบที่ทำงานตลอดเวลาที่ระบบทำงาน(undead tasks หรือ sombie) เช่น

Tasks: 95 total, 1 running, 94 sleeping, 0 stopped, 0 zombie

Cpu(s) แสดงเปอร์เซ็นการทำงานของซีพียูในโหมดการทำงานต่างๆ ของ OS เช่น user mode, system เป็นต้น

Cpu(s): 1.1%us, 2.6%sy, 0.1%ni, 93.2%id, 2.5%wa, 0.3%hi, 0.2%si, 0.1%st

Mem แสดงสถิติการใช้งานหน่วยความจำหลัก(main memory) ส่วนที่แสดงประกอบไปด้วยหน่วยความจำทั้งหมดที่ให้สามารถใช้ได้(total), หน่วยความจำที่เหลือ(free), หน่วยความจำที่ถูกใช้ไปแล้ว(used) และหน่วยความจำที่ใช้เป็นบัฟเฟอร์ชั่วคราว(buffers) เช่น

Mem: 476160k total,   416456k used,   59704k free,   50940k buffers

Swap แสดงสถิติการใช้งานหน่วยความจำ swap เช่น

Swap: 1048568k total,   0k used, 1048568k free,   273952k cached

PID หมายเลขโพรเซส(เลขจำนวนเต็ม)

PPID หมายเลขโพรเซสแม่หรือโพรเซสแรกที่สร้างโพรเซสลูกๆ เพิ่มเติมภายหลัง

UID หมายเลขผู้ใช้ที่สร้างโพรเซส

USER ชื่อผู้ที่สร้างโพรเซส

RUSER ชื่อจริงของผู้ที่สร้างโพรเซส

GROUP หมายเลขกลุ่มที่ผู้ใช้งานสังกัดอยู่

PR ลำดับความสำคัญ

NI nive value

nFLT จำนวน page fault(เกิดจากการค้นหาข้อมูลไม่พบในหน่วยความจำหลัก)

RES ขนาดของงานชนิดฝังตัว

VIRT ผลรวมของหน่วยความจำเสมือน(virtual memory) ที่ถูกใช้งาน

SHR ขนาดของหน่วยความจำที่ใช้งานร่วมกัน(ที่ถูกแบ่งไปใช้ในแต่ละโพรเซส)

TIME เวลาที่ต้องใช้ในการประมวลผลแต่ละงาน

TIME+ เหมือนกับ TIME

%CPU ปริมาณการใช้งานซีพียูของแต่ละโพรเซส

%MEM ปริมาณการใช้งานหน่วยความจำของแต่ละโพรเซส

COMMAND คำสั่งที่กำลังทำงาน

ตัวอย่างการใช้งาน :

```
# top
top - 21:13:06 up  4:10,  2 users,  load average: 0.03, 0.08, 0.04
Tasks:  97 total,   1 running,  96 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  2.6%sy,  0.0%ni, 95.7%id,  0.0%wa,  1.6%hi,  0.0%si,  0.0%st
Mem:    476160k total,   415296k used,    60864k free,    52844k buffers
Swap:  1048568k total,        0k used,  1048568k free,   269700k cached
```

```
 PID USER     PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
6921 root     15   0  2204 1004  796 R  1.0  0.2   0:00.24 top
4861 haldaemo 18   0  5720 3728 1624 S  0.7  0.8   0:08.75 hald
4880 root     16   0  1960  628  548 S  0.7  0.1   0:26.46 hald-addon-stor
6739 root     15   0  9856 2776 2240 S  0.3  0.6   0:00.74 sshd
   1 root     15   0  2076  640  548 S  0.0  0.1   0:02.17 init
   2 root     RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
   3 root     34  19     0    0    0 S  0.0  0.0   0:00.02 ksoftirqd/0
   4 root     RT  -5     0    0    0 S  0.0  0.0   0:00.04 watchdog/0
```

แสดงการใช้คำสั่ง top โดยไม่มีออพชัน เมื่อเข้าสู่โปรแกรม top แล้วผู้ใช้สามารถตอบโต้กับ top ได้แบบทันที ซึ่งคำสั่งที่ใช้งานสามารถอ่านได้จากคำอธิบายด้านบน หรือจะใช้คำสั่ง h เพื่อขอดูสรุปคำสั่งก็ได้ เมื่อต้องการออกจาก top ให้กดปุ่ม q หรือ Ctrl + c

```
# top –u rpc
PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4342 rpc      18   0  1816  544  452 S  0.0  0.1   0:00.02 portmap
```

แสดงรายงานเฉพาะโพรเซสที่สั่งด้วยผู้ใช้ชื่อว่า rpc เท่านั้น

```
# top -p 140, 3811
PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 140 root     25   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
3811 root     11  -5     0    0    0 S  0.0  0.0   0:00.00 kjournald
```

แสดงรายงานเฉพาะโพรเซสหมายเลข 140 และ 3811 เท่านั้น

ลองทดสอบ :

```
# top -n 2
# top -b -n 1
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *ps, uptime*

คำสั่ง free

Processes management : **free**	
คำสั่ง	**free** เป็นคำสั่งที่ใช้สำหรับแสดงสถานะของหน่วยความจำหลัก เนื้อที่ว่างของหน่วยความจำ ทางกายภาพ, swap, หน่วยความจำที่ใช้งานร่วมกัน และบัฟเฟอร์
Syntax	free [*options*]
Example	# free # free -k

option :

-b,-k,-m,-g แสดงข้อมูลหน่วยความจำที่ใช้มีหน่วยเป็น ไบต์(b), กิโลไบต์(k), เม็กกะไบต์(m), กิกกะไบต์(g)

-s time ตรวจสอบหน่วยความจำที่ใช้ตามเวลาที่กำหนดใน time(มีหน่วยเป็นวินาที)

-t แสดงผลรวมของหน่วยความจำทั้งหมดที่ใช้งาน รวมกับ swap ด้วย

--help คำสั่งช่วยเหลือการใช้งาน

-V แสดงเวอร์ชันของ free

ตัวอย่างการใช้งาน :

```
# free
             total       used       free     shared    buffers     cached
Mem:        476160     381936      94224          0      26468     279556
-/+ buffers/cache:      75912     400248
Swap:      1048568          0    1048568
```

แสดงข้อมูลการใช้หน่วยความจำในระบบ total หมายถึงหน่วยความจำทั้งหมดที่มีในระบบ, used หมายถึงหน่วยความจำที่ถูกใช้ไปทั้งหมด, free หน่วยความจำที่เหลือ(free = total - used) ข้อมูลที่รายงานจะมีหน่วยเป็นไบต์

free -k

รายงานการใช้หน่วยความจำที่ใช้ โดยรายงานข้อมูลมีหน่วยวัดเป็นกิโลไบต์

free -s 2

รายงานการใช้หน่วยความจำที่กำลังใช้งาน ซึ่งจะ update ทุกๆ 2 วินาที โดยข้อมูลถูกรายงานมีหน่วยวัดเป็นไบต์

free -sk 2

รายงานการใช้หน่วยความจำที่กำลังใช้งาน ซึ่งจะ update ทุกๆ 2 วินาที โดยรายงานข้อมูล มีหน่วยวัดเป็นกิโลไบต์

free -tk

รายงานการใช้หน่วยความจำทั้งหมดที่กำลังใช้งาน โดยรายงานข้อมูล มีหน่วยวัดเป็นกิโลไบต์

ลองทดสอบ :

```
# free -mto
# cat /proc/meminfo
```

คำสั่งอื่นที่ทำงานคล้ายกัน :*top, cat /proc/meminfo*

คำสั่ง kill

Processes management : **kill**	
คำสั่ง	kill เป็นคำสั่งที่ใช้สำหรับยกเลิกการทำงานของโพรเซส หรือโปรแกรม ที่กำลังทำงาน
Syntax	kill [*options*] [*pids* | *commands*]
Example	# kill 1234 # kill -SIGHUP # kill -9 1234

option :

-a ยกเลิกโพรเซสที่ต้องการ (ต้องอ้างอิงที่อยู่ของคำสั่งแบบเต็ม เช่น /bin/kill –a gcc)

-l แสดงสัญญาณ(signals) เพื่อยกเลิกโพรเซสในรูปแบบต่างๆ เช่น SIGHUP, SIGKILL เป็นต้น

-s SIGNAL, -SIGNAL ยกเลิกการทำงานของโพรเซส โดยการระบุสัญญาณด้วย(สัญญาณดูได้จากออพชัน -l)

-9 ใช้ยกเลิกการทำงานของโพรเซสใด ๆ ที่ไม่สามารถยกเลิกได้

ตัวอย่างการใช้งาน :

kill -l

```
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL
 5) SIGTRAP     6) SIGABRT     7) SIGBUS      8) SIGFPE
 9) SIGKILL    10) SIGUSR1    11) SIGSEGV    12) SIGUSR2
13) SIGPIPE    14) SIGALRM    15) SIGTERM    16) SIGSTKFLT
17) SIGCHLD    18) SIGCONT    19) SIGSTOP    20) SIGTSTP
…
```

แสดงสัญญาณ(signal) ที่บอกให้คำสั่ง kill ทราบว่าจะยกเลิกการทำงานของโพรเซสในรูปแบบใด

ps –aux

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START
TIME COMMAND
root         1  0.0  0.1   2080   640 ?        Ss   15:22   0:04 init [5]
```

รายงานโพรเซสที่กำลังทำงานอยู่ในระบบทั้งหมดอย่างละเอียด เนื่องจากคำสั่ง kill มีความจำเป็นต้องรู้ PID ของโพรเซสที่ต้องการยกเลิกก่อน

kill 3302

[1]+ Terminated find / -name "*test"

สั่งยกเลิกโพรเซสที่เกิดจากคำสั่ง find ตัวเลข 3302 หมายถึง หมายเลขโพรเซส(PID)

kill -s SIGTERM 1234 หรือ **kill -15 1234**

ยกเลิกการทำงานของโพรเซสหมายเลข 1234 โดยใช้สัญญาณ SIGTERM (SIGTERM เป็นการบอกให้โพรเซสที่กำลังจะถูกยกเลิก มีการบันทึกข้อมูล เช่น ไฟล์คอนฟิกหรือข้อมูลอื่นๆ ที่จำเป็นก่อน โดยปกติคำสั่ง kill จะใช้สัญญาณนี้เสมอ) หรือสามารถใช้ตัวเลขระบุสัญญาณก็ได้(-15 คือ SIGTERM)

kill -s SIGKILL 1234

ยกเลิกการทำงานของโพรเซสหมายเลข 1234 โดยใช้สัญญาณ SIGKILL (SIGKILL หรือ -9 เป็นการบอกให้โพรเซสที่กำลังทำงาน ยกเลิกการทำงานทันทีโดยไม่มีการบันทึกข้อมูลใดๆ จะใช้ในกรณีที่คำสั่ง kill แบบปกติไม่สามารถยกเลิกการทำงานของโพรเซสได้)

```
# kill -KILL 1234
```
ยกเลิกการทำงานของโพรเซสเหมือน kill –s SIGKILL 1234
```
# kill -9 1234
```
ยกเลิกการทำงานของโพรเซสเหมือน kill –s SIGKILL 1234

ลองทดสอบ :

```
# kill 8939 9543
# kill -SIGKILL 1414
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *bg, csh, fg, jobs, ksh, ps*

คำสั่ง nice, renice

Processes management : **nice**	
คำสั่ง	**nice** เป็นคำสั่งที่ใช้สำหรับตั้งค่า(nice) และปรับ(renice) ระดับความสำคัญให้โพรเซส
Syntax	**nice [*option*] [*command* [*arguments*]], renice [*priority*] [*options*] [*target*]**
Example	**# nice -8 top** **# nice -10 top** **# renice 19 2243**

option :

-n adjustment, -adjustment, --adjustment=adjustment สั่งเพิ่มหรือลดลำดับความสำคัญของโพรเซสที่ทำงานในระบบ โดยปกติโพรเซสที่ทำงานเป็นปกติจะมีค่าเท่ากับ 10 สำหรับค่าที่สามารถปรับได้จะมีช่วงตั้งแต่ 1-19 (ตัวเลขที่มีค่าต่ำจะมีลำดับความสำคัญสูงกว่า)

-p ระบุหมายเลขโพรเซสที่ต้องการเพิ่ม/ลดความสำคัญ

-help คำสั่งอธิบายคำสั่ง nice

ตัวอย่างการใช้งาน :

```
# nice -13 vi myfile.txt
```
ปรับลำดับความสำคัญของคำสั่ง vi ให้ลดลง ค่าปกติของ vi เดิมเป็น 10
(สำหรับแสดงค่าลำดับความสำคัญของโพรเซสจะใช้คำสั่ง ps)
```
# ps -aux
0    0 5773 5743 28 13  4924 1320 -    SN+ pts/1    0:00
vi myfile.txt
```
แสดงผลของการปรับลำดับของโพรเซสด้วยคำสั่ง nice -13
```
# renice 16 -p 13245
```
ปรับลำดับความสำคัญของโพรเซสหมายเลข 12345 ให้มีค่าเท่ากับ 16

ลองทดสอบ :

```
# nice --10 ./top &
# ps axl | grep top
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *csh, ksh, nohup, priocntl, sh*

5.3.15 กลุ่มคำสั่งเกี่ยวกับ Scheduling Jobs

คำสั่ง sleep

Scheduling Jobs : **sleep**	
คำสั่ง	**sleep** เป็นคำสั่งที่ใช้สำหรับพักการทำงานของโพรเซสชั่วคราว เริ่มตั้งแต่ 1 วินาทีไป จนถึงหลายๆ วัน
Syntax	**sleep *amount*[*units*] *sleep option***
Example	**# sleep 10**

option :

-help คำสั่งอธิบายคำสั่ง sleep

-version คำสั่งแสดงเวอร์ชันของ sleep

ตัวอย่างการใช้งาน :

sleep 1
สั่งให้โพรเซสหยุดพักการทำงาน 1 วินาที เมื่อครบเวลาแล้วจะกลับเข้ามาทำงานในระบบอีกครั้ง(คำสั่ง sleep นิยมให้ทำงานร่วมกับคำสั่งอื่นๆ เมื่อต้องการรอคอยอะไรบางอย่างทำงานเสร็จก่อน)

sleep 3 &
สั่งให้โพรเซสหยุดพักการทำงาน 3 วินาที โดยทำงานเป็นแบบ background process (ใช้เครื่องหมาย &)

sleep 180
สั่งให้โพรเซสหยุดพักการทำงาน 3 นาที

jobs
[1]+ Running sleep 120 &
เป็นคำสั่งที่ใช้ตรวจสอบโพรเซสที่หยุดพักอยู่ในระบบ

ลองทดสอบ :

sleep 1200 &

คำสั่งอื่นที่ทำงานคล้ายกัน : *date, time, wait*

คำสั่ง watch

Scheduling Jobs : **watch**	
คำสั่ง	**watch** เป็นคำสั่งที่ใช้สำหรับสั่งให้โปรแกรมทำงานซ้ำๆ ในระยะเวลาที่กำหนด(ค่า default คือ 2 วินาที)
Syntax	**watch [*options*] *command* [*cmd_options*]**
Example	**# watch free -m**

option :

-d, --differences[=cumulative] ระบุให้แสดงรายงานการเปลี่ยนแปลงในแต่ละรอบของการทำงาน เมื่อจุดใดที่รายงานมีการเปลี่ยนแปลงจะทำสัญญลักษณ์คือ highlight(แถบสีเทา) ตรงข้อมูลที่เกิดการเปลี่ยนแปลง

-help คำสั่งอธิบายคำสั่ง watch

-n secs, --interval=secs สั่งรันคำสั่งตามเวลาที่กำหนด มีหน่วยเป็นวินาที

-t, --no-title ไม่แสดงส่วนหัวที่เกิดจากคำสั่ง watch

-version คำสั่งแสดงเวอร์ชันของ watch

ตัวอย่างการใช้งาน :

```
# watch -n 5 free -m
Every 5.0s: free -m      Sat Feb 27 02:14:31 2010
         total     used      free    shared   buffers    cached
Mem:       465       457       7        0       50       318
-/+ buffers/cache:       87       377
Swap:      1023        0     1023
```

สั่งให้คำสั่ง free รายงานหน่วยความจำที่ใช้งาน ในทุกๆ 5 วินาที โดยใช้คำสั่ง watch ควบคุมเวลาในการทำงาน

```
# watch -n 5 –t free -m
         total     used      free    shared   buffers    cached
Mem:       465       457       7        0       50       318
-/+ buffers/cache:       87       377
Swap:      1023        0     1023
```

สั่งให้คำสั่ง free รายงานหน่วยความจำที่ใช้งาน ในทุกๆ 5 วินาที เหมือนตัวอย่างด้านบนแต่ไม่ต้องแสดงส่วนหัวที่เกิดจากคำสั่ง watch คือข้อความ Every 5.0s: free -m Sat Feb 27 02:14:31 2010 โดยใช้ -t

```
# watch -d ls –l
Every 2.0s: ls -l    Sat Feb 27 02:24:20 2010
total 0
-rw-r--r-- 1 root root 0 Feb 26 18:52 myfile
-rw-r--r-- 1 root root 0 Feb 26 18:52 myfile.txt
```

สั่งให้คำสั่ง ls –l ทำงานทุกๆ 2 วินาที ต่อจากนั้นเวลาจะถูกนับเพิ่มไปเรื่อยๆ ตามจำนวนการทำงานในแต่ละรอบ ในที่นี้จะเวลาจะเพิ่มขึ้นรอบละ 2 วินาที

```
# watch netstat –tn
Every 2.0s: netstat -tn   Sat Feb 27 02:33:03 2010
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address  Foreign Address    State
tcp    0     0 ::ffff:192.168.1.4:22  ::ffff:192.168.1.2:4687
ESTABLISHED
```

สั่งให้คำสั่งตรวจสอบการเชื่อมต่อและการเปิดพอร์ต(netstat) ทำงาน

ทุกๆ 2 นาที

```
# watch ifconfig eth0
Every 2.0s: ifconfig eth0   Sat Feb 27 02:34:52 2010
eth0      Link encap:Ethernet  HWaddr 00:0C:29:0A:5A:E9
          inet addr:192.168.1.4  Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe0a:5ae9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:1008 errors:0 dropped:0 overruns:0 frame:0
          TX packets:707 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:106067 (103.5 KiB)  TX bytes:100902 (98.5 KiB)
```

ตรวจสอบการทำงานและอัตราการส่งข้อมูลของการ์ดเน็ตเวิร์ค(ifconfig eth0) ที่ชื่อว่า eth0 ทุกๆ 2 วินาที

```
# watch --differences ifconfig eth0
Every 2.0s: ifconfig eth0   Sat Feb 27 02:34:52 2010
eth0      Link encap:Ethernet  HWaddr 00:0C:29:0A:5A:E9
          inet addr:192.168.1.4  Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe0a:5ae9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:1164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:849 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:125160 (122.2 KiB)  TX bytes:117730 (114.9 KiB)
```

ตรวจสอบการทำงานของการ์ดเน็ตเวิร์คเหมือนตัวอย่างข้างบน แต่จะ highlight ส่วนที่มีการเปลี่ยนแปลงเกิดขึ้นด้วย

ลองทดสอบ :

```
# watch echo $$
# watch echo '$$'
# watch -n2 "ps aux|grep http"
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *at, crontab*

คำสั่ง at

Scheduling Jobs : **at**	
คำสั่ง	**at** เป็นคำสั่งที่ใช้สำหรับสั่งให้โปรแกรม, คำสั่ง หรือเชลล์สคริปต์ทำงานซ้ำๆ ตามระยะเวลาที่กำหนด
Syntax	**at [*options*] *time* [*date*]**
Example	**# at -f myjob.sh 10:25**

option :

-d job [job...] ลบงานหรือ job ที่สั่งด้วยคำสั่ง at ออกจากคิว

-f file อ่านคำสั่งจากไฟล์ เช่น เชลล์สคริปต์

-l แสดงรายงานของคำสั่งที่ได้สร้างไว้

-m เมลล์แจ้งไปยังชื่ออีเมล์ที่กำหนด หลังจากทำงานเสร็จสิ้นแล้ว

-q letter กำหนดลำดับการทำงานของโพรเซสในคิว ซึ่งคิวจะถูกจัดเรียงตามตัวอักษร a-z ตัวอักษรที่อยู่ท้าย จะมีลำดับความสำคัญต่ำไปเรื่อยๆ(ค่า z จะต่ำที่สุด)

-help คำสั่งอธิบายคำสั่ง at

การกำหนดเวลา(Time)

hh:[mm] [modifiers] ผู้ใช้สามารถกำหนดเวลาในการทำงานได้ โดยกำหนดที่ตัวแปล hh (ชั่วโมง) โดยใช้ 24 ชั่วโมงเป็นค่า default, mm(นาที) มีค่าตั้งแต่ 0-59 ตัวอย่างเช่น 5:30 (hh=05, mm=30) ในส่วนของ modifiers จะเป็นสัญญลักษณ์ที่เพิ่มเข้ามาเพื่อบอกความหมายของเวลาในรูปแบบอื่นๆ เช่น pm, am หมายถึงกลางวันหรือกลางคืน zulu หมายถึงเวลาเทียบกับ Greenwich เป็นต้น

การกำหนดวันที่(Date)

month num[, year] ผู้ใช้สามารถกำหนดวันที่ที่ต้องการสั่งงาน โดย month คือเดือนมีค่าตั้งแต่ 1-12 หรือถ้าต้องการระบุเป็นตัวอักษรให้กำหนดเพียง 3 ตัวแรกของชื่อเดือนก็ได้เช่น jan หมายถึง janurary, num คือวันที่ มีค่าตั้งแต่ 1-31 และ year คือปี มีจำนวน 4 หลักเช่น 2010

today | tomorrow คำสั่ง at ยังมีความสามารถให้ผู้ใช้กำหนดเวลาในรูปแบบอื่นๆ ได้ เช่น today หมายถึงวันนี้ หรือ tormorrow หมายถึงพรุ่งนี้เป็นต้น



NOTE: กำหนดให้วันนี้เป็นวันอังคาร(tuesday) ที่ 18 กันยาน (september) 2010 เวลา 10:00 AM จากตัวอย่างจะใช้คำสั่ง at ในรูปแบบต่างๆ ดังนี้

ตัวอย่างคำสั่ง	คำอธิบาย
at noon	12:00 PM September 18, 2010
at midnight	12:00 AM September 19, 2010
at teatime	4:00 PM September 18, 2010
at tomorrow	10:00 AM September 19, 2010
at noon tomorrow	12:00 PM September 19, 2010
at next week	10:00 AM September 25, 2010
at next monday	10:00 AM September 24, 2010
at fri	10:00 AM September 21, 2010

at OCT	10:00 AM October 18, 2010
at 9:00 AM	9:00 AM September 19, 2010
at 2:30 PM	2:30 PM September 18, 2010
at 1430	2:30 PM September 18, 2010
at 2:30 PM tomorrow	2:30 PM September 19, 2010
at 2:30 PM next month	2:30 PM October 18, 2010
at 2:30 PM Fri	2:30 PM September 21, 2010
at 2:30 PM 9/21	2:30 PM September 21, 2010
at 2:30 PM Sept 21	2:30 PM September 21, 2010
at 2:30 PM 9/21/2010	2:30 PM September 21, 2010
at 2:30 PM 9.21.10	2:30 PM September 21, 2010
at now + 30 minutes	10:30 AM September 18, 2010
at now + 1 hour	11:00 AM September 18, 2010
at now + 2 days	10:00 AM September 20, 2010
at 4 PM + 2 days	4:00 PM September 20, 2010
at now + 3 weeks	10:00 AM October 9, 2010
at now + 4 months	10:00 AM January 18, 2011
at now + 5 years	10:00 AM September 18, 2015

ตัวอย่างการใช้งาน :

Script myjob
ls -al
at 01:35 < myjob
job 1 at 2010-02-28 01:35

สั่งตั้งเวลาโดยใช้คำสั่ง at ให้สั่งสคริปต์ชื่อว่า myjob(มีคำสั่ง ls –al อยู่) ให้ทำงานตามเวลาที่กำหนด (สมมุติว่าวันนี้เป็นวันที่ 28 เดือนกุมภาพันธ์ 2010) เวลา 01:35 นาที job ดังกล่าวเป็น job แรกที่สร้างขึ้นดังนั้นจำมีลำดับการทำงานในคิวเป็นลำดับที่ 1

at -l
1 2010-02-28 01:35 a root

แสดงคำสั่งที่อยู่ในคิวที่เกิดจากคำสั่ง at สร้างขึ้นจากตัวอย่าง job 1 เป็นของผู้ใช้ root และมีชื่อ job เป็นตัวอักษรชื่อว่า a

at –r 1

ลบ job 1 หรือ a ออกจากคิว

```
# at -m 01:35 < myjob
job 2 at 2010-02-28 01:35
```

สั่งตั้งเวลาโดยใช้คำสั่ง at เหมือนคำสั่งข้างต้น แต่เมื่อคำสั่งดังกล่าวทำงานเรียบร้อยแล้วจะส่งอีเมลล์ไปยังผู้ใช้ที่สร้าง job ดังกล่าวขึ้นมาด้วย (ในที่นี้เมลล์จะส่งไปที่ root@localhost)

```
# at -f myjob.sh 10:25
job 6 at 2010-02-27 10:25
```

สั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh วันที่ 27 เดือนกุมภาพันธ์ 2010 เวลา 10:25 นาที

```
# at -f myjob.sh  10pm tomorrow
job 8 at 2010-02-28 22:00
```

สั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh วันที่ 28 เดือนกุมภาพันธ์ 2010 เวลา 22:00 นาที(สมมุติว่าวันนี้คือวันที่ 27 กุมภาพันธ์ 2010) 10pm คือเวลาที่เริ่มตั้งแต่เที่ยงวัน – เที่ยงคืน(เริ่มตั้งแต่บ่ายโมงตรงให้นับเป็น 1pm, 6 โมงเย็น= 6pm ดังนั้น 10pm คือ 22:00 น) คำว่า tomorrow หมายถึงวันพรุ่งนี้

```
# at -f myjob.sh 2:00 tuesday
job 9 at 2010-03-02 02:00
```

สั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh วันที่ 2 เดือนมีนาคม 2010 เวลา 2:00 นาที(สมมุติว่าวันนี้คือวันเสาร์ที่ 27 กุมภาพันธ์ 2010) และนับต่อไปอีก 3 วันซึ่งตรงกัยวันที่ 2 มีนาคม

```
# at -f myjob.sh 2:00 july 11
job 10 at 2010-07-11 02:00
```

สั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh วันที่ 11 เดือนกรกฎาคม 2010 เวลา 2:00 นาที

```
# at -f myjob.sh 2:00 next week
job 11 at 2010-03-06 02:00
```

สั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh วันที่ 6 เดือนมีนาคม 2010 เวลา 2:00 นาที

ลองทดสอบ :

```
# atq
# at now + 5 minutes
# at 4pm + 3 days
# at 10am Jul 31
# at 1750 sep 17
# at 5:50pm sep 17
# at 4 am Sunday
# at now + 6 hours
# at midnight tomorrow
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *crontab*

คำสั่ง crontab

Scheduling Jobs : **crontab**	
คำสั่ง	**crontab** เป็นคำสั่งที่ใช้สำหรับกำหนดเวลาในการสั่งประมวลผลคำสั่ง, เชลล์สคริปต์ , หรือโปรแกรม ตามเวลาที่กำหนด
Syntax	**crontab [*options*] [*file*]**
Example	**# crontab -e** **# crontab -l**

option :

-e edit ใช้ในการเรียกโปรแกรม เท็กซ์อิดิตเตอร์สำหรับ อ่านข้อความที่เขียนไว้ในไฟล์ crontab ซึ่งใช้ในกรณีที่ผู้ใช้งาน ต้องการเพิ่มเติมหรือลด บรรทัด ชุดคำสั่งต่างๆ

-l list ใช้ในการแสดงเนื้อที่หาภายในไฟล์ crontab ของผู้ใช้รายนั้นๆ

-r remove ใช้ในการยกเลิกและลบไฟล์ crontab

-i ยืนยันก่อนการลบ crontab

-s ใช้งานร่วมกับ selinux

-u ระบุรายชื่อผู้ใช้งาน crontab

-help คำสั่งอธิบายคำสั่ง crontab

รูปแบบในการสั่งงาน crontab

นาที (Min)	**ชั่วโมง (Hour)**	**วันภายในเดือน (DayOfMonth)**	**เดือนภายในปี (MonthOfYear)**	**วันภายในสัปดาห์ (DayOfWeek)**	**คำสั่ง (Command)**
0	12	13	2	*	Tar –cvf bkup.tar /

จากตัวอย่างข้างต้น crontab แบ่งออกเป็น 2 ส่วนคือ

ส่วนที่ 1(นาที) ใช้สำหรับกำหนดเวลาในส่วนของนาที โดยค่าที่สามารถกำหนดได้เริ่มตั้งแต่ 0-59

ส่วนที่ 2(ชั่วโมง) ใช้สำหรับกำหนดเวลาในส่วนของชั่วโมง โดยค่าที่สามารถกำหนดได้เริ่มตั้งแต่ 0-23 (0 เท่ากับเที่ยงคืน)

ส่วนที่ 3(วันภายในเดือน) ใช้สำหรับกำหนดวันที่ในแต่ละเดือน เช่น เดือนมกราคมมี 31 วัน เดือนเมษายนมี 30 วัน เป็นต้น โดยค่าที่สามารถกำหนดได้เริ่มตั้งแต่ 1-31

ส่วนที่ 4(เดือนภายในปี) ใช้สำหรับกำหนดเดือนภายใน 1 ปีซึ่งมีทั้งหมด 12 เดือน โดยเดือนแรกคือเดือนมกราคม โดยค่าที่สามารถกำหนดได้เริ่มตั้งแต่ 1-12

ส่วนที่ 5(วันภายในสัปดาห์) ใช้สำหรับกำหนดวันใน 1 สัปดาห์ โดยเริ่มวันแรกคือวันอาทิตย์มีค่าเป็น 0 โดยค่าที่สามารถกำหนดได้เริ่มตั้งแต่ 0-6

ส่วนที่ 6(คำสั่ง) ใช้สำหรับระบุถึงคำสั่งที่ต้องการให้ทำงาน



NOTE: รูปแบบในการสั่งงาน crontab พิเศษเพิ่มเติม

* (wild card) กำหนดให้ crontab ทำงานทุกๆ ช่วงเวลา

*/5 กำหนดให้ crontab ทำงานทุกๆ 5 นาทีเมื่ออยู่ในส่วนที่ 1 และทำงานทุกๆ 5 ชั่วโมงถ้าปรากฏอยู่ในส่วนที่ 2 หรือทำงานทุกๆ 5 วัน เมื่อปรากฏในส่วนที่ 3 เป็นต้น

2, 4, 6 แทนความหมายว่า หรือ(OR) เมื่อปรากฏในส่วนที่ 3 ของ crontab จะหมายถึงให้ทำงานวันที่ 3, 4 และ 6 ตามลำดับ

9-17 แทนความหมายว่าให้ทำงาน เริ่มตั้งแต่-สิ้นสุด เช่น ถ้าค่าดังกล่าวไปปรากฏที่ส่วนที่ 3 ของ crontab จะหมายถึง ให้ทำงานตั้งแต่วันที่ 9 ถึง วันที่ 17 ของเดือน

ตัวอย่างการใช้งาน :

ก่อนสั่งให้ crontab ทำงานจะต้องมีความเข้าใจการใช้ vi ในเบื้องต้นก่อน (อ่านเพิ่มเติมการใช้งานโปรแกรม vi) เริ่มต้นปรับแต่ง crontab โดยใช้คำสั่ง crontab –e โปรแกรมจะเข้าสู่ vi จากนั้นให้ผู้ใช้พิมพ์คำสั่ง เมื่อจบคำสั่งให้กดปุ่ม enter เพื่อเริ่มคำสั่งใหม่

* * * * * ls -l

สั่งให้คำสั่ง ls –l ทำงานทุกๆ 1 นาที

30 * * * * <command>

สั่งให้ทำงานทุกๆ 30 นาที (<command> คือคำสั่งอะไรก็ได้ที่ผู้ใช้งานต้องการสั่งให้ระบบทำงานให้)

45 6 * * * <command>

สั่งให้ทำงานทุกๆ วัน ที่เวลา 6 (AM) นาฬิกา 45 นาที (AM คือช่วงเวลาเริ่มตั้งแต่ หลังเที่ยงคืน – เที่ยงวัน และ PM คือช่วงเวลาหลังเที่ยงวัน - เที่ยงคืน)

45 18 * * * <command>

สั่งให้ทำงานทุกๆ วัน ที่เวลา 6 (PM) นาฬิกา 45 นาที

00 1 * * 0 <command>

สั่งให้ทำงานทุกๆ วันอาทิตย์ เวลา 1 (AM) นาฬิกา 0 นาที

00 1 * * 7 <command>

สั่งให้ทำงานทุกๆ วันอาทิตย์ เวลา 1 (AM) นาฬิกา 0 นาที

00 1 * * Sun <command>

```
สั่งให้ทำงานทุกๆ วันอาทิตย์ เวลา 1 (AM) นาฬิกา 0 นาที
# 30 8 1 * * <command>
สั่งให้ทำงานทุกๆ วันที่ 1 ของทุกๆ เดือน เวลา 8 (AM) นาฬิกา 30 นาที
# 00 0-23/2 02 07 * <command>
สั่งให้ทำงานชั่วโมงที่ 2 ของวันที่ 2 เดือนกรกฎาคม
```

ลองทดสอบ :

```
# 30 18 * * * rm /home/someuser/tmp/*
# 0 4 15-21 * 1 /command
# * * * * * /sbin/ping -c 1 192.168.0.1 > /dev/null
# 0 0,12 1 */2 * /sbin/ping -c 192.168.0.1; ls -la >>/var/log/cronrun
# 0 2 1-10 * * du -h --max-depth=1 /
# 20,50 * * Jan-May,7-12 Mon-Fri cd desktop/fold4"; ./foldlaunchscript
# 12 * * * * cd "desktop/fold2"; ./foldlaunchscript
# 1 10 13 * 5 ./hiddenfolder/jokescript
# 14 2 29 8 5 ./skynet
# 0,5,10,15,20,25,30,35,40,45,50,55 * * * * /command/to/execute
# 00 18 * * * /command/to/execute
# 30 08 10 06 * /home/ramesh/full-backup
# 00 11,16 * * * /home/ramesh/bin/incremental-backup
# 00 09-18 * * * /home/ramesh/bin/check-db-status
# 00 09-18 * * 1-5 /home/ramesh/bin/check-db-status
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *at*

5.3.16 กลุ่มคำสั่งเกี่ยวกับ Hosts

คำสั่ง hostname

Hosts : **hostname**	
คำสั่ง	**hostname** เป็นคำสั่งที่ใช้สำหรับแสดงหรือกำหนดชื่อโฮสต์ที่ใช้งานในปัจจุบัน
Syntax	**hostname [*option*] [*nameofhost*]**
Example	**# hostaname**

option :

-a, --alias แสดงชื่อที่ใช้แทน

-d, --domain แสดงชื่อโดเมนเนม

-f, --fqdn, --long แสดงชื่อโดเมนแบบยาว

-i, --ip-address แสดงหมายเลขที่อยู่หรือไอพี

-s, --short แสดงชื่อโดเมนแบบสั้น

-help คำสั่งอธิบายคำสั่ง hostname

ตัวอย่างการใช้งาน :

```
# hostname
Master.mus.ac.th
```

```
แสดงชื่อเครื่องปัจจุบันที่ใช้งาน
# hostname -d
mus.ac.th
แสดงชื่อโดเมนของเครื่องปัจจุบันที่ใช้งาน
# hostname -i
127.0.0.1
แสดงหมายเลขไอพีของเครื่องปัจจุบันที่ใช้งาน
# hostname -a
Master localhost.localdomain localhost
แสดงหมายชื่อที่ใช้เรียกแทน
```

ลองทดสอบ :

```
# hostname -f
# hostname -s
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *hostid, uname*

คำสั่ง ifconfig

Hosts : **ifconfig**	
คำสั่ง	**ifconfig** เป็นคำสั่งที่ใช้สำหรับแสดงหรือกำหนดค่าเกี่ยวกับการ์ดเน็ตเวิร์คอินเทอร์เฟส
Syntax	**ifconfig [*interface*] ifconfig [*interface address_family parameters addresses*]**
Example	**# ifconfig -a**

option :

พารามิเตอร์

คำสั่ง ifconfig จำเป็นต้องเรียกใช้งานผ่านชื่อเน็ตเวิร์คการ์ด เช่น ne0, eth0, lr0 และตามด้วยพารามิเตอร์ต่างๆ ดังต่อไปนี้

add address/prefixlength คำสั่งให้เพิ่มหมายเลขที่อยู่(ไอพี)ให้กับเน็ตเวิร์คอินเทอร์เฟสการ์ด

address address กำหนดหมายเลขไอพีให้กับเน็ตเวิร์คการ์ด

allmulti/-allmulti สั่งให้ทำงานหรือยกเลิกการส่งสัญญาณแจ้งไปยังเคอร์นอลในระดับชั้นทรานสพอร์ตเลเยอร์เมื่อมีเฟรมข้อมูลเข้า

arp/-arp สั่งให้ทำงานหรือยกเลิกการใช้โพรโทคอล arp ในการแปลงข้อมูลระหว่าง network-level addresses และ link-level addresses

hw กำหนดหมายเลขการ์ดเน็ตเวิร์ค หรือ MAC Address ใหม่

broadcast [address] กำหนดค่า broadcast ให้กับการ์ดเน็ตเวิร์ค

debug/-debug สั่งให้ทำงานหรือยกเลิกโหมดการ debug

del address/prefixlength ลบหมายและไอพีและซับเน็ตออกจากการ์ดเน็ตเวิร์ค

down สั่งให้การ์ดเน็ตเวิร์คหยุดทำงาน

irq addr กำหนดหมายเลขอินเทอร์ลัพธ์ให้กับการ์ดเน็ตเวิร์ค

metric n กำหนดขนาดของ เราท์เมตทริคให้กับการ์ดเน็ตเวิร์ค(ค่า default เป็น 0)

media type กำหนดชนิดของสื่อ เช่น 10base2, 10baseT เมื่อกำหนดให้เป็น auto คำสั่ง ifconfig จะพยายามหาสื่อที่เหมาะสมเอง

mtu n กำหนดขนาดของอัตราการส่งข้อมูลสูงสุดที่จะส่งในแต่ละครั้ง Maximum Transfer Unit (MTU)

multicast กำหนดว่าเป็นการสื่อสารแบบมัลติคลาสต์

netmask mask กำหนดค่าของเน็ตมาสค์ ค่าดังกล่าวเป็นค่าที่บ่งบอกว่าเครือข่ายดังกล่าวมีขนาดเท่าใด และช่วยบอกให้ทราบว่าข้อมูลจะเดินทางไปยังเครือข่ายดังกล่าวได้อย่างไร

pointopoint/-pointopoint [address] กำหนดรูปแบบการเชื่อมต่อแบบ จุดต่อจุด

txqueuelen n กำหนดขนาดของคิวให้กับการ์ดเน็ตเวิร์ค

tunnel addr สร้างอุโมงค์ ใช้ในกรณีที่ต้องการรัน IPV6 ไปบน IPV4

up สั่งให้ การ์ดเน็ตเวิร์คทำงาน

-a แสดงรายละเอียดข้อมูลของการ์ดเน็ตเวิร์ค

-help คำสั่งอธิบายคำสั่ง ifconfig

ตัวอย่างการใช้งาน :

```
# ifconfig -a
eth0     Link encap:Ethernet  HWaddr 00:0C:29:0A:5A:E9
         inet addr:192.168.1.4  Bcast:192.168.1.255
Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fe0a:5ae9/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
         RX packets:1057 errors:0 dropped:0 overruns:0 frame:0
         TX packets:820 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:106774 (104.2 KiB)  TX bytes:105243 (102.7
KiB)
```

แสดงข้อมูลรายละเอียดเกี่ยวกับการ์ดเน็ตเวิร์คอย่างละเอียด เช่น หมายเลข MAC Address(00:0C:29:0A:5A:E9), หมายเลขไอพีที่ใช้งาน (192.168.1.4), หมายเลขบอร์ดคลาสต์(192.168.1.255),เน็ตมาสค์ (255.255.255.0), MTU(1500), สถานะการทำงาน(UP) เป็นต้น

```
# ifconfig eth0
```

คำสั่งให้แสดงข้อมูลเฉพาะของการ์ด eth0 เท่านั้น

```
# ifconfig eth0 down
```

คำสั่งให้การ์ด eth0 หยุดการทำงาน

ifconfig eth0 up

คำสั่งให้การ์ด eth0 เริ่มต้นการทำงานใหม่

ifconfig eth0:1 192.168.1.10 netmask 255.255.255.0

```
eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:0A:5A:E9
          inet addr:192.168.1.10  Bcast:192.168.1.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
```

สั่งเพิ่มหมายเลขไอพีแอดเดรสตัวที่ 2 ไปยังการ์ดเน็ตเวิร์ค eth0 มีหมายเลขไอพีคือ 192.168.1.10 เน็ตมาร์ค 255.255.255.0 (หรือเรียกอีกชื่อว่า การสร้างซับอินเทอร์เฟส คือมีหลายไอพีบนการ์ดเน็ตเวิร์คใบเดียวกัน)

ifconfig eth0 hw ether 01:02:03:04:05:06

กำหนดหมายเลข MAC Address ใหม่ให้กับการ์ด etho ใหม่เป็น 01:02:03:04:05:06

ifconfig eth0 192.168.1.102 netmask 255.255.255.0 broadcast 192.168.1.255

กำหนดหมายเลขไอพีแอดเดรสให้กับการ์ดเน็ตเวิร์ค eth0 มีหมายเลขไอพีคือ 192.168.1.120 เน็ตมาร์ค 255.255.255.0 และบอร์ดคาซท์คือ 192.168.1.255

ลองทดสอบ :

```
# ifconfig eth0 mtu 1412
# ifconfig eth0 promisc
# ifconfig eth0 192.168.0.12 up
# ifconfig eth0 192.168.99.14 netmask 255.255.255.0 up
```

คำสั่งอื่นที่ทำงานคล้ายกัน *arp, dhclient, ifdown, ifup, ping, netstat, route*

คำสั่ง host

Hosts : **host**	
คำสั่ง	**host** เป็นคำสั่งที่ใช้สำหรับสืบค้นชื่อและหมายเลขไอพีของโฮสต์ในระบบโดเมนเนมเซิร์ฟเวอร์(DNS)
Syntax	**host [*options*] *name* [*server*]**
Example	**# host www.google.co.th**

DNS ย่อมาจาก Domain Name System Domain เป็นระบบจัดการแปลงชื่อไปเป็นหมายเลข IP address โดยมีโครงสร้างฐานข้อมูลแบบลำดับชั้นเพื่อใช้เก็บข้อมูลที่เรียกค้นได้อย่างรวดเร็ว

กลไกหลักของระบบ DNS คือ ทำหน้าที่แปลงข้อมูลชื่อเป็นหมายเลข IP address หรือแปลงหมายเลข IP address เป็นข้อมูลชื่อ เช่น ทำการแปลงข้อมูลจาก www.google.co.th เป็น 216.239.61.104 เป็นต้น

option :

พารามิเตอร์

-a เหมือนกับการใช้ออพชัน –v –t

-c class ระบุขอบเขตุในการค้นหาข้อมูลของ Dns เช่น สืบค้นเฉพาะ record ที่เป็น IN, CH, CHAOS, HS, HESIOD, หรือค้นหาทั้งหมดคือ ANY

-C รายงาน SOA record และ authorative name servers

-d คล้ายกับออพชัน –v

-l แสดงรายการในโดเมนทั้งหมด

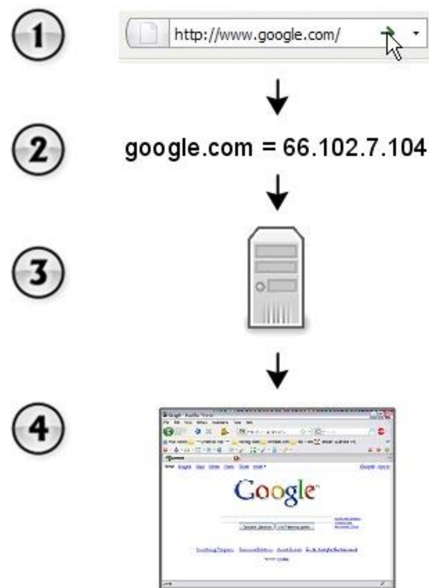
-t type ระบุชนิดของ recode ที่ต้องการสอบถามกับ dns เช่น CNAME, NS, SOA, SIG, หรือ ANY

-v แสดงรายละเอียดทั้งหมดในการแสดงผล เช่น time-to-live, class, authoritative nameservers เป็นต้น



NOTE: ขั้นตอนการทำงานของ DNS

1. ผู้ใช้งานต้องเข้าใช้งานเว็บให้บริการที่ตั้งอยู่บนเครือข่าย อินเทอร์เน็ต ผู้ใช้เริ่มต้นโดยใช้คำสั่ง host www.google.com โปรแกรม host จะทำการสอบถามไปยัง dns ขนาดเล็กที่เก็บในเครื่องของตนเองก่อน (สำหรับในกรณีของลินุกซ์จะเก็บอยู่ใน /etc/resolv.conf)
2. เมื่อไม่สามารถค้นหาที่อยู่ใน dns ในเครื่องของตนเองได้แล้ว จึงทำการร้องขอไปยังเครื่องเซิร์ฟเวอร์ที่ให้บริการ dns (เมื่อเครื่องที่ให้บริการไม่สามารถค้นหาได้อีกจะสอบถามต่อไปยัง root ของ dns ที่เปิดให้บริการอยู่ทั่วโลก [35])
3. ค้นหาข้อมูลในฐานข้อมูลใน dns เช่น www.google.com มี ไอพีคือ 66.102.7.104 กลับไปให้ผู้เรียกใช้งาน
4. นำไอพีที่ได้จาก dns เชื่อมต่อไปยังเซิร์ฟเวอร์ที่ให้บริการ(www.google.com) จริงบนเครือข่ายอินเทอร์เน็ต



ตัวอย่างการใช้งาน :

```
# host www.google.co.th
www.google.co.th is an alias for www.google.com.
www.google.com is an alias for www.l.google.com.
www.l.google.com has address 74.125.95.106
www.l.google.com has address 74.125.95.147
www.l.google.com has address 74.125.95.99
www.l.google.com has address 74.125.95.103
www.l.google.com has address 74.125.95.104
www.l.google.com has address 74.125.95.105
```

สอบถาม dns ว่า URL www.google.com มีหมายเลขไอพีอะไร

```
# host 202.28.32.42
42.32.28.202.in-addr.arpa domain name pointer www.msu.ac.th.
```

สอบถาม dns ว่าไอพี 202.28.32.42 มี URL คือ www.msu.ac.th

```
# host -C msu.ac.th
Nameserver nsinternal.msu.ac.th:
        msu.ac.th has SOA record nsinternal.msu.ac.th.
msuadmin.msu.ac.th. 3236705815 10800 3600 2592000 900
```

สอบถาม dns ว่า msu.ac.th ปรากฏอยู่ในฐานข้อมูลใดและมีข้อมูลการลงทะเบียนไว้เป็นอย่างไรบ้าง

ลองทดสอบ :

```
# host -l msu.ac.th
```

คำสั่งอื่นที่ทำงานคล้ายกัน *dig, nslookup, ping*

คำสั่ง whois

Hosts : **whois**

คำสั่ง	**whois** เป็นคำสั่งที่ใช้สำหรับสืบค้นข้อมูลการจดทะเบียนโดเมน
Syntax	**whois [*option*] *query***
Example	**# whois www.google.co.th**

option :

-h HOST, --host=HOST ระบุชื่อของเครื่องที่ต้องการสอบถาม เช่น msu.ac.th

-p ระบุหมายเลขพอร์ตที่ต้องการสอบถาม

-h เป็นคำสั่งช่วยเหลือการทำงานของ whois

ตัวอย่างการใช้งาน :

```
# whois msu.ac.th
[Querying whois.thnic.net]
[whois.thnic.net]

Whois Server Version 1.5

Domain: MSU.AC.TH
ACE: MSU.AC.TH
Registrar: T.H.NIC Co., Ltd.
Name Server: NS.MSU.AC.TH
Name Server: NS1.MSU.AC.TH
Status: ACTIVE
Updated Date: 15 Jan 2010
Created Date: 17 Jan 1999
Renew Date: 17 Jan 2010
Exp Date: 16 Jan 2020
Registrant: Mahasarakham University (
มหาวิทยาลัย มหาสารคาม )
        Mahasarakham UniversityTumbol khamreangAmphur
KantarawichaiMahasarakham44150,
Mahasarakham
        44150
        TH

Tech Contact: 11563
PORAR WEB APPLICATION CO., LTD.
9/5 Ratchadaphisek 18, Ratchadaphisek Road, Huaykwang,
Huaykwang, Bangkok
10310
TH
```

สอบถามรายละเอียดการข้อมูลการจดทะเบียนของโดเมนชื่อว่า msu.ac.th ข้อมูลที่แสดงจะบอกถึงรายละเอียดต่างๆ ของโดเมนดังกล่าว เช่น ชื่อโดเมน, จดกับหน่วยงานใด, วันที่จดโดเมน, วันหมดอายุ, ชื่อ DNS, ชื่อผู้จดทะเบียน, ที่อยู่ของผู้จดทะเบียนเป็นต้น

ลองทดสอบ :

```
# whois -h HOST google.com
```

คำสั่งอื่นที่ทำงานคล้ายกัน *who*

คำสั่ง ping

Hosts : **ping**	
คำสั่ง	**ping** เป็นคำสั่งที่ใช้สำหรับทดสอบการตอบสนองของโฮสต์ปลายทาง
Syntax	**ping [*options*] *host***
Example	**# ping www.google.com**

option :

-c count ระบุจำนวนครั้งของการตอบสนองจากโฮสต์ปลายทาง เมื่อครบจำนวนที่กำหนด คำสั่ง ping จะหยุดการส่งแพ็กเก็ตในการทดสอบ

-f สั่งให้คำสั่ง ping ทำการ flood ข้อมูลไปยังโฮสต์ปลายทางอย่างรวดเร็ว คือประมาณ 100 แพ็กเก็ตต่อวินาที (การทดสอบในลักษณะดังกล่าวไม่สมควรกระทำบ่อย เนื่องจากอาจจะถูกมองว่าเป็นการโจมตีเครือข่ายแทน)

-i wait ระบุเวลาการรอคอยของแต่ละแพ็กเก็ตที่เดินทาง โดยปกติแต่ละแพ็กเก็ตจะเดินทางห่างกันประมาณ 1 วินาที

-R บันทึกและพิมพ์เส้นทางที่เกิดจาก ECHO_REQUEST และ ECHO_REPLY

-s packetsize ระบุขนาดของแพ็กเก็ตที่ต้องการทดสอบ ค่าโดย default จะมีขนาดเท่ากับ 64 ไบต์ ประกอบด้วยเฮดเดอร์ของโพโทคอล ICMP 8 ไบต์ กับข้อมูลอีก 56 ไบต์

-t n กำหนดเวลาของ IP Time to Live มีหน่วยเป็นวินาที

-w n กำหนดเวลาให้โปรแกรม ping หยุดทำงานหลังจากหมดเวลาที่กำหนดในตัวแปร n

-h เป็นคำสั่งช่วยเหลือการทำงานของ ping

ตัวอย่างการใช้งาน :

```
# ping www.google.com
PING www.l.google.com (64.233.181.147) 56(84) bytes of data.
64 bytes from www.google.com (64.233.181.147): icmp_seq=1
ttl=49 time=67.4 ms
64 bytes from www.google.com (64.233.181.147): icmp_seq=2
ttl=49 time=74.7 ms
64 bytes from www.google.com (64.233.181.147): icmp_seq=3
ttl=49 time=63.7 ms
64 bytes from www.google.com (64.233.181.147): icmp_seq=4
ttl=49 time=126 ms
64 bytes from www.google.com (64.233.181.147): icmp_seq=5
ttl=49 time=61.9 ms
```

```
64 bytes from www.google.com (64.233.181.147): icmp_seq=6
ttl=49 time=71.2 ms
64 bytes from www.google.com (64.233.181.147): icmp_seq=7
ttl=49 time=62.8 ms
--- www.l.google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6006ms
rtt min/avg/max/mdev = 61.915/75.485/126.460/21.264 ms
```

ทดสอบการ ping ไปยังโฮสต์ www.google.com จากตัวอย่างจะเห็นว่า google.com มีหมายเลขไอพีคือ 64.233.181.147 ข้อมูลที่ได้จากคำสั่ง ping จะแสดงรายงานของอัตราการส่งแพ็คเก็ต, ค่าเฉลี่ยการตอบสนอง, ความเร็วของการตอบสนองที่ได้รับสูงสุด เป็นต้น

ping -c 1 www.google.com

```
PING www.google.com (64.233.181.147) 56(84) bytes of data.
64 bytes from www.google.com (64.233.181.147): icmp_seq=1
ttl=49 time=59.5 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 59.591/59.591/59.591/0.000 ms
```

สั่งให้ทำการ ping www.google.com เพียง 1 แพ็กเก็ตเท่านั้น จากนั้นให้รายงานผล

ping -i 0.1 www.google.com

สั่งให้ทำการ ping www.google.com โดยกำหนดให้แพ็กเก็ตที่ส่งไปห่างกันประมาณ 0.1 วินาที

ลองทดสอบ :

```
# ping 0
# ping localhost
# ping 127.0.0.1
# ping -f localhost
# ping -s 100 localhost
# ping -w 5 localhost
# ping -c 4 0 -w 2
# ping -R 192.168.1.63
```

คำสั่งอื่นที่ทำงานคล้ายกัน *host, ifconfig, netstat, rpcinfo, traceroute*

คำสั่ง traceroute

Hosts : **traceroute**	
คำสั่ง	**traceroute** เป็นคำสั่งที่ใช้สำหรับตรวจสอบเส้นทางจากเครื่องที่ทำการทดสอบไปสู่โฮสต์ปลายทาง
Syntax	**traceroute [*options*] *host* [*packetsize*]**
Example	**# traceroute www.google.com**

option :

ตัวอย่างการใช้งาน :

```
# traceroute www.google.com
traceroute to www.google.co.th (64.233.181.105), 30 hops max,
40 byte packets
 1  mygateway1.ar7 (192.168.1.1)  2.959 ms  2.640 ms  3.087 ms
 2  114.128.216 (114.128.216.1)  29.602 ms  27.567 ms  28.846 ms
 3  * * *
 4  * * *
 5  mx (58.147.0.41)  39.726 ms  38.936 ms  39.682 ms
 6  * * *
 7  mx-ll-58.147.0-69.static.tttmaxnet.com (58.147.0.69)
42.950 ms  35.904 ms  35.202 ms
 8   (218.100.47.246)  36.414 ms  35.664 ms  36.307 ms
 9   (218.100.47.22)  37.851 ms  36.136 ms  36.004 ms
10   (72.14.217.193)  61.152 ms  60.419 ms  61.192 ms
11   (209.85.242.238)  62.093 ms  62.130 ms  62.517 ms
...
16  www.google.co.th (64.233.181.105)  60.383 ms  63.822 ms
62.252 ms
```

ทดสอบเส้นทางจากเครื่องผู้ใช้งานไปยังโฮสต์ www.google.com จากตัวอย่างจะเห็นว่าเส้นทางเริ่มต้นที่โหนด mygateway1.ar7 จากนั้นเดินทางต่อไปยังโหนด 114.128.216.1 ไปเรื่อยๆ ตามลำดับ สุดท้ายจะถึงโหนดที่ 16 คือหมายเลขไอพี 64.233.181.105 ซึ่งเป็นโฮสต์ google.com นั่นเอง สังเกตุเห็นว่าจำนวนของเส้นทางอาจจะไม่เหมือนกันในการสั่ง traceroute แต่ละครั้ง ทั้งนี้เนื่องจากเส้นทางบนเครือข่ายอินเทอร์เน็ตจะมีการเปลี่ยนแปลงอยู่เสมอ

ลองทดสอบ :

```
# traceroute 0
```

คำสั่งอื่นที่ทำงานคล้ายกัน *netstat, ping*

5.3.17 กลุ่มคำสั่งเกี่ยวกับ Networking

คำสั่ง ssh

Networking : **ssh**	
คำสั่ง	**ssh** เป็นคำสั่งที่ใช้สำหรับเข้าใช้งานเครื่องโฮสต์จากระยะไกล (ข้อมูลที่รับส่งทั้งหมดจะมีการเข้ารหัส)
Syntax	**ssh [*options*] *hostname* [*command*]**
Example	**# ssh 192.168.1.10**

option :

-l login_name ระบุชื่อที่ใช้สำหรับ login

-p port ระบุพอร์ตที่ต้องการเชื่อมต่อไปยังโฮสต์ปลายทาง

ตัวอย่างการใช้งาน :

ssh -l suchart remotehost.example.com

ทำการเชื่อมต่อไปยังโฮสต์ปลายทางชื่อว่า remotehost.example.com โดยระบุชื่อผู้ใช้ด้วยคำสั่ง –l เมื่อเชื่อมต่อสำเร็จโปรแกรมจะถามรหัสผ่าน ให้ผู้ใช้ใส่รหัสผ่านที่ถูกต้อง เมื่อเข้าสู่ระบบได้แล้วการใช้งานคำสั่งจะเหมือนกับการใช้งานผ่าน command line ตามปกติทั่วไป (การใช้คำสั่ง ssh โฮสต์ปลายทางจะต้องทำการติดตั้ง ssh เซิร์ฟเวอร์ก่อน)

ssh 192.168.1.100

ทำการเชื่อมต่อไปยังโฮสต์โดยการระบุหมายเลขไอพีโดยตรง

คำสั่ง telnet

ใช้ติดต่อเข้า server ต่าง ๆ ตาม port ที่ต้องการ แต่ปัจจุบัน server ต่าง ๆ ปิดบริการ telnet แต่เปิด SSH แทน

telnet 202.202.202.202 :: ขอติดต่อเข้าเครื่อง 202.202.202.202 การไม่กำหนด port คือเข้า port 23

telnet www.school.net.th 21 :: ขอติดต่อผ่าน port 21 ซึ่งเป็น FTP port

telnet mail.loxinfo.co.th 25 :: ตรวจ smtp ว่าตอบสนองกลับมา หรือไม่

telnet class.yonok.ac.th 110 :: ทดสอบ pop service ของ windows server 2003

telnet 202.29.78.13 80 :: ให้พิมพ์คำสั่ง GET แม้มองไม่เห็นหลังกดปุ่ม enter (ใช้ทดสอบการตอบสนองของ server)

คำสั่ง scp

สำเนาไฟล์ ระหว่างโฮสต์(มีการเข้ารหัสข้อมูล)

scp *.txt user@remote.server.com:/home/user/

สำเนาไฟล์ทุกไฟล์ที่มีส่วนขยาย .txt ไปยังโฮสต์ remote.server.com ด้วยผู้ใช้ชื่อว่า user และมี home ไดเรคทรอรีอยู่ที่ /home/user

คำสั่ง sftp

บริการ โอนถ่ายไฟล์ระหว่างโฮสต์(มีการเข้ารหัสข้อมูล)

เชื่อมต่อไปยัง sftp server ชื่อว่า shell.example.com

sftp shell.example.com

Connecting to shell.example.com...

Password:

sftp>

คำสั่ง ftp

บริการโอนถ่าย ไฟล์ระหว่างโฮสต์(ไม่มีการเข้ารหัสข้อมูล)

```
# ftp ftp.example.com
```

คำสั่ง mutt

โปรแกรมใช้งานอี เมล์แบบ text

```
# mutt -f /var/mail/user
```

คำสั่ง mail

คำสั่งรับส่งอีเมล์ ผ่าน command line

```
# mail -f /var/mail/user
# mail goofy@domain.com
```

คำสั่ง mozilla

โปรแกรมเว็บบราวเซอร์แบบ กราฟฟิก

คำสั่ง lynx

โปรแกรมเว็บบราว เซอร์แบบ text

```
# lynx www.google.com
```

คำสั่ง wget

ดาวน์โหลดข้อ มูลเว็บมาสู่ดิสก์

```
# wget -c http://www.example.com/large.file
# wget -nd -r -l1 --no-parent http://www.foo.com/mp3/
```

คำสั่ง talk

คำสั่งรับส่งข้อความโต้ ตอบ

```
# talk suchart
```

คำสั่ง write

คำสั่งส่ง ข้อความไปยังจอภาพอื่น

```
# write aizatto pts/0
```


5.3.18 กลุ่มคำสั่งเกี่ยวกับ System Information [31]

เป็นกลุ่มของคำสั่งที่แสดงรายละเอียดของข้อมูลระบบ ซึ่งประกอบด้วยคำสั่งต่อไปนี้

คำสั่ง arch

System information : **arch**	
คำสั่ง	arch แสดงข้อมูลสถาปัตยกรรมของระบบ
Syntax	arch *[-k | archname]*
Example	# arch

option :

-k แสดงสถาปัตยกรรมของลินุกซ์เคอร์เนลที่กำลังทำงานอยู่ เช่น i386, i686, sun4n

ตัวอย่างการใช้งาน :

```
# arch
# arch -k
686
ผลลัพธ์ที่ได้คือเคอร์เนลของลินุกซ์ที่กำลังทำงานอยู่เป็นแบบ x86
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *uname, ps*

คำสั่ง date

System information : **date**	
คำสั่ง	**date** แสดงและกำหนดเวลาให้กับระบบ
Syntax	**date [*options*] [+*format*] [*date*]**
Example	**# date** **# date +"%d-%m-%Y"** **# date -s "16:15:00"**

options : -

-d date หรือ --date date ซึ่ง date จะอยู่ในรูปของ format ที่กำหนดเองได้ และควรอยู่ในเครื่องหมาย ' ' หรือ " " สำหรับรูปแบบที่สามารถกำหนดได้ เช่น Yesterday, tomorrow, last month, next month, 5 days ago, 7 days, next Saturday, 2 years 3 months 4 day

-f datefile, --file=datefile คล้ายการทำงานด้วย –d แต่ datefile จะอยู่ในรูปของไฟล์แทน เช่น ไฟล์ชื่อว่า date.txt มีข้อมูลคือ

5 days ago

7 days

next Saturda

จากนั้นให้รันด้วยคำสั่ง date -f date.txt หรือ date –file=date.txt

-I [timespec] , --iso-8601[=timespec] แสดงข้อมูลในลักษณะของ ISO-8601 format

-r file, --reference=file แสดงข้อมูลของไฟล์ที่ถูกแก้ไขครั้งล่าสุด

-R, --rfc-822 แสดงข้อมูลใน format ของ RFC 822

--help คำสั่งช่วยเหลือการใช้งาน

--version แสดงเวอร์ชันของคำสั่ง

-s date, --set date กำหนดเวลาให้กับระบบ

-u, --universal กำหนดเวลาตาม Greenwich Mean Time (ไม่ใช่เวลาท้องถิ่น)

+Format : แสดงวันเวลาที่ไม่ใช่รูปแบบมาตรฐาน รูปแบบที่กำหนดจะต้องมีเครื่องหมาย + ด้านหน้าเสมอ รูปแบบในการแสดงผลจะอยู่ในเครื่องหมาย ' ' หรือ " " และต้องมีสัญลักษณ์นำหน้าคือ %, -, _ ตัวอย่างเช่น %a

%a แสดงชื่อวันแบบย่อ เช่น tue = Tuesday

%b แสดงชื่อเดือนแบบย่อ เช่น jan = january

%c กำหนดรูปแบบวันเวลาแบบเจาะจงประเทศ

%d กำหนดวันของเดือน มีค่าตั้งแต่ 01-31

%h เหมือน %b

%j Julian day of year (001-366)

%k แสดงผลแบบ 1 วันมี 24 ชั่วโมง(0-23)

%l แสดงผลแบบ 1 วันมี 12 ชั่วโมง(1-12)

%m แสดงผลเดือนในแต่ละปี(01-12)

%n เพิ่มบรรทัดใหม่ 1 บรรทัด

%p แสดงเวลาแบบ am หรือ pm

%s ระบุเวลาตามรูปแบบ 1970-01-01 00:00:00 UTC

%t เพิ่มแทบใหม่

%w วันในอาทิตย์(Sunday = 0)

%x ระบุเวลาของแต่ละประเทศในรูปแบบท้องถิ่น

%y กำหนดขนาดการแสดงผล 2 ตำแหน่งของปี

%A แสดงชื่อของวันแบบเต็ม เช่น Tuesday

%B แสดงชื่อเต็มของเดือน

%D แสดงวันเวลาในรูปแบบ %m/%d/%y

%H แสดงข้อมูลแบบ 24 ชั่วโมง (00-23)

%I แสดงข้อมูลแบบ 12 ชั่วโมง (01-12)

%M แสดงนาที (00-59)

%S วินาที (00-59)

%T แสดงวันเวลาในรูปแบบ %H:%M:%S

%U จำนวนของสัปดาห์ในแต่ละปี เริ่มวันอาทิตย์ในแต่ละสัปดาห์ (00-53)

%Y แสดงปีแบบ 4 หลัก เช่น 2010

%Z แสดงชื่อ Time Zone

ตัวอย่างการใช้งาน :

```
# date
Tue Jan 12 12:31:14 ICT 2010
```
แสดงวันที่ เวลา โดยอ้างอิงเวลาท้องถิ่น(ค่า default)
```
# date +"%d"
12
```
แสดงเฉพาะวันของเดือน ในตัวอย่างคือวันที่ 12
```
# date +"%d%m%y"
120110
```
แสดงวันที่ ในรูปแบบ วัน เดือนและปีติดกัน(ปี ค.ศ. แสดง 2 หลัก)
```
# date +"%a%d%m%y"
Tue120110
```
แสดงวันที่โดย แสดงวันของสัปดาห์แบบย่อด้วย(tue=tuesday) ต่อด้วย วัน เดือน ปี ติดกัน
```
# date +"%d-%m-%Y"
12-01-2010
```
แสดงวันที่ ในรูปแบบ วัน เดือน ปี โดยมี - คั่นไว้
```
# date +"%d/%m/%Y"
12/01/2010
```
แสดงวันที่ ในรูปแบบ วัน เดือน ปี โดยมี / คั่นไว้
```
# date +"%A,%B %d %Y"
Tuesday,January 12 2010
```
แสดงชื่อวันอังคารแบบเต็ม คั่นด้วย , เดือนมกราคมแบบเต็ม วันที่ 12 และปี ค.ศ. 4 หลัก
```
# date --date="yesterday"
Mon Jan 11 12:33:42 ICT 2010
```
แสดงวัน เวลา ที่ผ่านมาแล้วเมื่อวาน จากตัวอย่างวันนี้เป็นวันที่ 12 เดือนมกราคม 2010 เมื่อวานคือ 11 มกราคม 2010
```
# date --date="tomorrow"
Wed Jan 13 12:33:55 ICT 2010
```
แสดงวัน เวลา วัดถัดไป จากตัวอย่างวันนี้เป็นวันที่ 12 เดือนมกราคม 2010 วันถัดไปคือ 13 มกราคม 2010
```
# date --date="last month"
Sat Dec 12 12:34:07 ICT 2009
```
แสดงวัน เวลา ของเดือนที่ผ่าน(ปัจจุบันคือ 12/01/2010 เดือนที่แล้วคือ 12/12/2009)
```
# date --date="next month"
Fri Feb 12 12:34:20 ICT 2010
```
แสดงวัน เวลา ของเดือนถัดไป(ปัจจุบันคือ 12/01/2010 เดือนถัดไปคือ 12/02/2010)
```
# date --date="5 days ago"
Thu Jan  7 12:34:31 ICT 2010
```
ปัจจุบันคือ 12/01/2010 ผ่านมาแล้ว 5 วันคือวันที่ 07/01/2010
```
# date --date="7 days"
```

```
Tue Jan 19 12:34:41 ICT 2010
ปัจจุบันคือ 12/01/2010 อีก 7 วันคือวันที่ 19/01/2010
# date --date='next Saturday'
Sat Jan 16 00:00:00 ICT 2010
ปัจจุบันคือ วันอังคารที่ 12/01/2010 วันเสาร์ที่จะมาถึงคือ 16/01/2010
# date --date='2 years 3 months 4 day'
Mon Apr 16 12:35:07 ICT 2012
ปัจจุบันคือวันอังคาร ที่ 12/01/2010 อีก 2 ปี 3 เดือน 4 วันคือวันจันทร์ ที่ 16/04/2012
# date -s "16:15:00"
Tue Jan 12 16:15:00 ICT 2010
กำหนดเวลาให้กับระบบ เป็น 16 นาฬิกา 15 นาที 0 วินาที
# date -s "16:55:30 July 7, 1986"
Mon Jul  7 16:55:30 ICT 1986
กำหนดเวลาให้กับระบบ เป็นวันที่ 7 กรกฎาคม ค.ศ. 1986 เวลา 16 นาฬิกา 55 นาที 30 วินาที
# date 033121422003.55
Mon Mar 31 21:42:55 ICT 2003
กำหนดเวลาให้กับระบบเป็นแบบตัวเลข 03=เดือนมีนาคม(Mar), 31=วันที่ 31, 2142= 21 นาฬิกา 42 นาที, 2003=ปี ค.ศ., .55 วินาที
หมายเหตุ: การกำหนดเวลาดังกล่าวใช้รูปแบบ MMDDhhmmCCYY.ss เมื่อต้องการบันทึกเข้าสู่ bios ให้ใช้คำสั่ง clock -w
```

ลองทดสอบ :

```
# date +"|%y|%m|%d"
# date +"[%y:%m:%d]"
# date -d "3 months 100 days"
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *time*

คำสั่ง cal

System information : **cal**	
คำสั่ง	**cal** แสดงปฏิทินในรูปแบบของเดือน และปี
Syntax	***# cal [-13smjyV] [[month] year]***
Example	**cal** **cal 10 2010**

option :

-1 แสดงปฏิทินของเดือนปัจจุบันเท่านั้น

-3 แสดงปฏิทิน 3 เดือนคือ เดือนปัจจุบัน เดือนก่อนหน้า และเดือนถัดไป

-s แสดงวันอาทิตย์เป็นวันแรกของสัปดาห์(default)

-m แสดงวันจันทร์เป็นวันแรกของสัปดาห์

-y แสดงปฏิทินในปีปัจจุบัน

-V เวอร์ชันของ cal

parameter :

month ระบุถึงเดือนที่ต้องการให้แสดง ต้องกำหนดเป็นเลขจำนวนเต็ม เช่น เดือนมกราคม มีค่าเป็น 1 และเดือนธันวาคมมีค่าเป็น 12

year ระบุถึงปีที่ต้องการให้แสดง

ตัวอย่างการใช้งาน

```
# cal
January 2010
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
แสดงปฏิทินเฉพาะเดือนปัจจุบัน
# cal 12 2009
December 2009
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
แสดงปฏิทินของเดือนธันวาคม ปี ค.ศ. 2009
```

ลองทดสอบ :

```
# cal -1
# cal -3
# cal -13 2010
# cal -3sm 9 2010
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *calendar*

คำสั่ง whoami

ใช้เพื่อแสดงว่าผู้ใช้ซึ่ง login เข้าสู่ระบบนั้น (ตัวเราเอง) login ด้วยชื่ออะไร

โครงสร้างคำสั่ง/ตัวอย่าง **# whoami**

คำสั่ง finger

ใช้สำหรับแสดงรายละเอียดของผู้ใช้

โครงสร้างคำสั่ง finger [user@host] หรือ finger [@host]

กรณีไม่ระบุชื่อ finger จะแสดงรายละเอียดของ User ที่กำลัง logon อยู่บนเครื่องนั้นๆ ทั้งหมด ซึ่งหากไม่ระบุ host ด้วย โปรแกรมจะถือว่าหมายถึงเครื่องปัจจุบัน

ตัวอย่าง **# finger**

คำสั่ง uname

System information : **uname**	
คำสั่ง	**uname** แสดงข้อมูลของระบบ
Syntax	**uname *[options]***
Example	**# uname -a**

options :

-a, --all แสดงข้อมูลพื้นฐานทั้งหมดของระบบ

-i แสดงข้อมูลของสถาปัตยกรรมที่ใช้

-m แสดงของมูลของฮาร์ดแวร์ที่ใช้

-n แสดงชื่อของเครื่อง

-p ใช้งานเหมือน –m (นิยมใช้มากกว่า –m)

-r แสดง release ของระบบปฏิบัติการ

-s แสดงชื่อของระบบปฏิบัติการที่ใช้

-v แสดงข้อมูล เวอร์ชันของระบบปฏิบัติการ

-o เหมือนกัน -s

ตัวอย่างการใช้งาน :

```
# uname -a
Linux localhost 2.6.18-92.el5 #1 SMP Tue Jun 10 18:49:47 EDT 2008 i686
i686 i386 GNU/Linux
```
แสดงรายละเอียดพื้นฐานของระบบทั้งหมด

```
# uname -snm
Linux localhost i686
```
แสดงรายละเอียดของระบบ โดยแสดงเฉพาะ ชื่อระบบปฏิบัติการที่ใช้ ชื่อเครื่อง และฮาร์ดแวร์ที่ใช้งาน

ลองทดสอบ :

```
# uname -ismr
```

คำสั่งอื่นที่ทำงานคล้ายกันหรือเกี่ยวข้อง : *arch, isalist, sysinfo*

คำสั่ง cat /proc/cpuinfo

System information : **/proc/cpuinfo**	
คำสั่ง	**cpuinfo** แสดงรายละเอียดของซีพียู(CPU)
Syntax	***/proc/cpuinfo***
Example	**# cat /proc/cpuinfo**
comment	cpuinfo ไม่ใช่คำสั่งแต่เป็นไฟล์ที่เก็บข้อมูลของ CPU

ตัวอย่างการใช้งาน :

```
# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 14
model name      : Genuine Intel(R) CPU T2130  @ 1.86GHz
stepping        : 8
cpu MHz         : 1862.237
cache size      : 1024 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 10
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr
pge mca cmov pat clflush dts acpi mmx fxsr sse sse2 ss nx
constant_tsc up pni
bogomips        : 3728.62
```

แสดงรายละเอียดของ cpu ที่กำลังใช้งานอยู่

ลองทดสอบ :

```
# more /proc/cpuinfo
# less /proc/cpuinfo
# head /proc/cpuinfo
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/meminfo

System information : **/proc/meminfo**	
คำสั่ง	**meminfo** แสดงรายละเอียดของหน่วยความจำ
Syntax	***/proc/meminfo***

Example	# cat /proc/meminfo
comment	meminfo ไม่ใช่คำสั่งแต่เป็นไฟล์ที่เก็บข้อมูลของหน่วยความจำ

ตัวอย่างการใช้งาน :

```
# cat /proc/meminfo
MemTotal:          515492 kB
MemFree:           52204 kB
Buffers:           25288 kB
Cached:            315344 kB
SwapCached:        0 kB
Active:            158972 kB
Inactive:          279736 kB
HighTotal:         0 kB
HighFree:          0 kB
LowTotal:          515492 kB
LowFree:           52204 kB
SwapTotal:         1048568 kB
SwapFree:          1048568 kB
Dirty:             20 kB
Writeback:         0 kB
AnonPages:         98080 kB
Mapped:            46696 kB
Slab:              15616 kB
PageTables:        3168 kB
NFS_Unstable:      0 kB
Bounce:            0 kB
CommitLimit:       1306312 kB
Committed_AS:      414088 kB
VmallocTotal:      507896 kB
VmallocUsed:       3736 kB
VmallocChunk:      503984 kB
HugePages_Total:   0
HugePages_Free:    0
HugePages_Rsvd:    0
Hugepagesize:      4096 kB
```

ลองทดสอบ :

```
# more /proc/meminfo
# less /proc/meminfo
# head /proc/meminfo
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/interrupts

System information : **/proc/interrupts**	
คำสั่ง	**interrupts** แสดงรายละเอียดของการอินเทอร์รัพท์
Syntax	*/proc/interrupts*

Example	# cat /proc/interrupts
comment	interrupts ไม่ใช่คำสั่งแต่เป็นไฟล์ที่เก็บข้อมูลของการอินเทอร์รัพท์ไว้

ตัวอย่างการใช้งาน :

```
# cat /proc/interrupts
        CPU0
  0:   2288797 IO-APIC-edge  timer
  1:       383 IO-APIC-edge  i8042
  6:         5 IO-APIC-edge  floppy
  7:         0 IO-APIC-edge  parport0
  8:         1 IO-APIC-edge  rtc
  9:         0 IO-APIC-level acpi
 12:     20091 IO-APIC-edge  i8042
 15:     20284 IO-APIC-edge  ide1
169:      5437 IO-APIC-level eth0
177:         0 IO-APIC-level ehci_hcd:usb1
185:         0 IO-APIC-level uhci_hcd:usb2, Ensoniq AudioPCI
193:     14391 IO-APIC-level ioc0
NMI:         0
LOC:   2288919
ERR:         0
MIS:         0
```

แสดงรายละเอียดของหมายเลขอินเทอร์รัพท์ ที่กำลังใช้งานอยู่ เช่น 6 คือหมายเลข interrupt ของ floppy disk

ลองทดสอบ :

```
# more /proc/interrupts
# less /proc/interrupts
# head /proc/interrupts
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/swaps

System information : **/proc/swaps**	
คำสั่ง	**swaps** แสดงรายละเอียดของ swap ไฟล์
Syntax	*/proc/swaps*
Example	# cat /proc/swaps
comment	swaps ไม่ใช่คำสั่งแต่เป็น swap ไฟล์

ตัวอย่างการใช้งาน :

```
# cat /proc/swaps
Filename                        Type        Size     Used  Priority
/dev/mapper/VolGroup00-LogVol01 partition   1048568  0     -1
```

ลองทดสอบ :

```
# more /proc/swaps
```

```
# less /proc/swaps
# head /proc/swaps
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/version

System information : **/proc/version**	
คำสั่ง	**version** แสดงรายละเอียดเวอร์ชันของลินุกซ์เคอร์เนล
Syntax	*/proc/version*
Example	# cat /proc/version
comment	version ไม่ใช่คำสั่งแต่เป็นไฟล์แสดงรายละเอียดเวอร์ชันของลินุกซ์เคอร์เนล

ตัวอย่างการใช้งาน :

```
# cat /proc/version
Linux version 2.6.18-92.el5 (mockbuild@builder16.centos.org) (gcc version
4.1.2 20071124 (Red Hat 4.1.2-42)) #1 SMP Tue Jun 10 18:49:47 EDT 2008
```

ลองทดสอบ :

```
# more /proc/version
# less /proc/version
# head /proc/version
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/net/dev

System information : **/proc/net/dev**	
คำสั่ง	**net/dev** แสดงรายละเอียดของการ์ดเน็ตเวิร์คและสถิติการใช้งาน
Syntax	*/proc/net/dev*
Example	# cat /proc/net/dev
comment	net/dev ไม่ใช่คำสั่งแต่เป็นไฟล์ข้อมูลแสดงรายละเอียดของการ์ดเน็ตเวิร์คและสถิติการใช้งาน

ตัวอย่างการใช้งาน :

```
# cat /proc/net/dev
Inter-|  Receive                                    | Transmit
 face |bytes   packets errs drop fifo frame compressed multicast|bytes   packets errs
drop fifo colls carrier compressed
   lo: 7577126    3253    0    0    0    0        0        0  7577126    3253    0    0    0    0
0        0
  eth0:  663657    6218    0    0    0    0        0        0   150513    1017    0    0    0    0
0        0
  sit0:       0       0    0    0    0    0        0        0        0       0    0    0    0    0
0
```

ลองทดสอบ :

```
# more /proc/net/dev
# less /proc/net/dev
# head /proc/net/dev
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/mounts

System information : **/proc/mounts**	
คำสั่ง	**mounts** แสดงรายละเอียดของไฟล์ที่ผูกเข้ากับลินุกซ์(mount)
Syntax	***/proc/mounts***
Example	**# cat /proc/mounts**
comment	mounts ไม่ใช่คำสั่งแต่เป็นไฟล์แสดงรายละเอียดของไฟล์ที่ปรากฏในลินุกซ์

ตัวอย่างการใช้งาน :

```
# cat /proc/mounts
rootfs / rootfs rw 0 0
/dev/root / ext3 rw,data=ordered 0 0
/dev /dev tmpfs rw 0 0
/proc /proc proc rw 0 0
/sys /sys sysfs rw 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
devpts /dev/pts devpts rw 0 0
/dev/sda1 /boot ext3 rw,data=ordered 0 0
tmpfs /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
/etc/auto.misc /misc autofs
rw,fd=6,pgrp=4524,timeout=300,minproto=5,maxproto=5,indirect 0 0
-hosts /net autofs
rw,fd=12,pgrp=4524,timeout=300,minproto=5,maxproto=5,indirect 0 0
```

จากตัวอย่าง ระบบมีการ mount ไฟล์ชื่อ /dev/root เป็นชนิด ext3 ชนิดที่สามารถเขียนและอ่านได้ เป็นต้น

ลองทดสอบ :

```
# more /proc/mounts
# less /proc/mounts
# head /proc/mounts
```

คำสั่งอื่นที่ทำงานคล้ายกัน : *more, less, head*

คำสั่ง cat /proc/fstab

สำหรับตรวจสอบ file system table เพื่อบอกว่ามีอะไร mount ไว้แล้วบ้าง

คำสั่ง clock

System information : **clock**	
คำสั่ง	**clock** แสดงเวลาของระบบ
Syntax	clock *[-w]*
Example	# clock

options :

-w บันทึกเวลาไว้ใน BIOS

ตัวอย่างการใช้งาน :

```
# clock
Tue 12 Jan 2010 11:07:33 AM ICT  -0.640110 seconds
# clock -w
```

การบันทึกเวลาด้วย option w จะใช้เมื่อมีการแก้ไขเวลาด้วยคำสั่ง date เสียก่อนและจึงบัททึกลงสู่ bios

คำสั่งอื่นที่ทำงานคล้ายกัน : *date*

คำสั่ง dmidecode

System information : **dmidecode**	
คำสั่ง	**dmidecode** แสดงรายละเอียดของฮาร์ดแวร์(DMI table decoder)
Syntax	dmidecode *[options]*
Example	# dmidecode

options :

-q, --quiet แสดงข้อมูลอย่างย่อ

-u, --dump แสดงแบบไบรารี่

-t, --type TYPE แสดงเฉพาะแต่ละ DIM Type เช่น Memory Controller

ตัวอย่างการใช้งาน :

```
# dmidecode -q
BIOS Information
        Vendor: Phoenix Technologies LTD
        Version: 6.00
        Release Date: 04/10/2007
        Address: 0xE7A00
        Runtime Size: 99840 bytes
        ROM Size: 64 kB
        Characteristics:
.....
```

ลองทดสอบ :

```
# dmidecode -u
# # dmidecode --type System Boot Information
```

คำสั่งอื่นที่ทำงานคล้ายกันหรือเกี่ยวข้อง : *biosdecode , mem , ownership , vpddecode*

คำสั่ง hdparm

System information : **hdparm**	
คำสั่ง	**hdparm** ตรวจสอบและแสดงคุณสมบัติของฮาร์ดดิสก์
Syntax	hdparm *[options] [device]*
Example	# hdparm –i /dev/hda1

options :

-C ตรวจสอบการทำงานของไฟฟ้าของฮาร์ดดิสก์

-i แสดงรายละเอียดของฮาร์ดดิสก์

parameter : -

device ชื่อของฮาร์ดดิสก์ เช่น IDE จะเป็น /dev/hda, SCSI เป็น /dev/sda

ตัวอย่างการใช้งาน :

```
# dhparm –i /dev/hda1
แสดงรายละเอียดของฮาร์ดดิสก์กรณีเป็น IDE
# dhparm –i /dev/sda1
แสดงรายละเอียดของฮาร์ดดิสก์กรณีเป็น SCSI และ SATA
```

ลองทดสอบ :

```
# dhparm -C /dev/hda1
# dhparm -g /dev/hda1
# dhparm -tT /dev/hda1
```

คำสั่งอื่นที่ทำงานคล้ายกันหรือเกี่ยวข้อง : -

คำสั่ง lspci

System information : **lspci**	
คำสั่ง	**lspci** แสดงข้อมูลการ์ดชนิด pci ที่เชื่อมต่อในระบบ
Syntax	lspci *[options]*
Example	# lspci

options :

-t แสดงข้อมูลบัสของ pci แบบทรี

-v แสดงรายละเอียดแบบยาว

ตัวอย่างการใช้งาน :

```
# lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX
Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP
bridge (rev 01)
```

```
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
...
```
แสดงรายละเอียดการ์ด pci ที่เชื่อมต่อกับระบบ

lspci -tv
```
-[0000:00]-+-00.0  Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge
           +-01.0-[0000:01]--
           +-07.0  Intel Corporation 82371AB/EB/MB PIIX4 ISA
           +-07.1  Intel Corporation 82371AB/EB/MB PIIX4 IDE
           +-07.2  Intel Corporation 82371AB/EB/MB PIIX4 USB
           +-07.3  Intel Corporation 82371AB/EB/MB PIIX4 ACPI
           +-0f.0  VMware Inc Abstract SVGA II Adapter
           +-10.0  LSI Logic / Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI
           \-11.0-[0000:02]--+-00.0  Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
                             +-01.0  Ensoniq ES1371 [AudioPCI-97]
                             \-02.0  VMware Inc Abstract USB2 EHCI Controller
```
แสดงรายละเอียดการ์ด pci ที่เชื่อมต่อกับระบบแบบโครงสร้างทรี

ลองทดสอบ :

lspci -nv
lspci -D

คำสั่งอื่นที่ทำงานคล้ายกันหรือเกี่ยวข้อง : *setpci, update-pciids*

คำสั่ง lsusb

System information : **lsusb**	
คำสั่ง	lsusb แสดงข้อมูลของอุปกรณ์ ยูเอสบี
Syntax	lsusb *[options]*
Example	# lsusb

options :

-t แสดงข้อมูลของ usb แบบทรี

-v แสดงรายละเอียดแบบยาว

ตัวอย่างการใช้งาน :

lsusb
```
Bus 001 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
```
แสดงรายละเอียดของ usb ที่เชื่อมต่อกับระบบ

lsusb -tv
```
Bus#  2
`-Dev#   1 Vendor 0x0000 Product 0x0000
```

```
Bus# 1
`-Dev# 1 Vendor 0x0000 Product 0x0000
```
แสดงรายละเอียด usb ที่เชื่อมต่อกับระบบแบบโครงสร้างทรี

ลองทดสอบ :

```
# lspci -D 001
```

คำสั่งอื่นที่ทำงานคล้ายกันหรือเกี่ยวข้อง : -

คำสั่ง man

เป็นคำสั่งที่สำคัญมาก เพราะจะช่วยให้อธิบายคำสั่งต่าง ๆ

ตัวอย่างคำสั่ง และการใช้งาน

man man เพื่ออธิบายคำสั่ง man เอง ว่าตัวคำสั่งนี้ใช้อย่างไร

man ls เพื่ออธิบายคำสั่ง ls ว่าใช้อย่างไร

man useradd เพื่ออธิบายคำสั่ง useradd ว่าใช้อย่างไร

5.3.19 กลุ่มคำสั่งเกี่ยวกับ System Maintenance [29-30]

คำสั่ง init

System Management : **init**	
คำสั่ง	**init, telinit** เป็นคำสั่งเริ่มต้นการทำงานของระบบปฏิบัติการ โดยปกติจะถูกสั่งจาก boot loader คือ LILO หรือ GRUB
Syntax	**init [*bootflags*] [*runlevel*]**
Example	**# init 0** **# init 5** **# init 6**

bootflags :

-a, auto เมื่อกำหนดให้ AUTOBOOT มีค่าเป็น yes boot loader จะอ่านค่าคอนฟิกทุกอย่างตามค่า default และจะให้ควบคุมการทำงานผ่าน command line เท่านั้น

-b ทำการ boot เครื่องและเข้าสู่โหมด single-user ใช้ในกรณีฉุกเฉิน

-s, S, single เข้าสู่โหมด single-user

-b, emergency เข้าสู่โหมด single-user แต่จะไม่สั่งรันสคริปต์ใดๆ ทั้งสิ้น

files :

init เป็นโพรเซสแรกขณะที่ระบบปฏิบัติการเริ่มการทำงาน ตรวจสอบความถูกต้องของโครงสร้างของไฟล์ระบบแล้ว init จะเริ่มสร้างโพรเซสลูกด้วยคำสั่ง fork และ exec ที่ระบุไว้ในไฟล์ /etc/inittab ซึ่งโพรเซสต่างๆ ที่สั่งให้ทำงานก็อาจจะถูกควบคุมผ่าน runlevel อีก

ครั้ง โพรเซสทั้งหมดที่จบการทำงานข้อมูลจะถูกเก็บไว้ใน */var/run/utmp* และ */var/log/wtmp*

runlevels :

การเปลี่ยนลำดับการทำงาน (runlevel) จะสามารถสั่งงานผ่านทาง telinit หรือ init ก็ได้ ซึ่งระดับการทำงานมีอยู่ด้วยกันหลายระดับคือ

-0 สั่งให้ระบบหยุดทำงาน (system halt)

-1, s, S เข้าสู่ single-mode ระบบจะใช้งานได้เพียงคนเดียวเท่านั้น ใช้สำหรับเข้าแก้ไขเมื่อระบบมีปัญหา

-3 เข้าสู่โหมดแบบเข้าใช้งานพร้อมกันได้หลายคน(Multiuser mode) การควบคุมการทำงานทั้งหมดจะกระทำผ่าน command line และนิยมใช้กรณีให้บริการเป็นเซิร์ฟเวอร์

-5 เข้าสู่ระบบแบบกราฟฟิก ใช้งานได้พร้อมกันหลายคน การควบคุมและสั่งงานผ่านกราฟฟิก

-6 สั่งให้ระบบเริ่มต้นทำงานใหม่ โดยปกติจะไม่กำหนดให้ใช้ใน inittab เพราะเครื่องจะ restart ตลอดเวลา

ตัวอย่างการใช้งาน :

```
# init 0
สั่ง shutdown ระบบและปิดเครื่อง
# init 1
สั่ง restart เครื่องใหม่ จากนั้นระบบจะเข้าสู่โหมด single-user
# init 3
สั่ง restart เครื่องใหม่ จากนั้นระบบจะเข้าสู่โหมด multi-user
# init 5
สั่ง restart เครื่องใหม่ จากนั้นระบบจะเข้าสู่โหมด multi-user + กราฟฟิก
# init 6
สั่ง restart เครื่องใหม่
```

ลองทดสอบ :

```
# init s
# init 9
```

ดูคำสั่งอื่นประกอบ : */etc/inittab*

คำสั่ง logout, exit

System Management : **logout, exit**	
คำสั่ง	**logout, exit** เป็นคำสั่งที่ต้องการออกจากเชลล์ หรือออกจาก remote เชลล์
Syntax	exit, logout
Example	# exit

	# logout

ตัวอย่างการใช้งาน :

logout สั่งออกจากเชลล์ หรือออกจากเชลล์ที่เราได้ทำการ remote ไปใช้งานยังเครื่องที่ตั้งอยู่บนเครือข่าย **# exit** สั่งออกจากเชลล์ หรือออกจากเชลล์ที่เราได้ทำการ remote ไปใช้งานยังเครื่องที่ตั้งอยู่บนเครือข่าย

ลองทดสอบ :

quit **# bye** **# ctrl + D** (กดปุ่ม ctrl พร้อมกับ d, D)

ดูคำสั่งอื่นประกอบ : *break, csh, ksh, sh, reboot, shutdown*

คำสั่ง reboot

System Management : **reboot**	
คำสั่ง	**reboot** เป็นคำสั่งที่ใช้เริ่มต้นระบบใหม่ การใช้คำสั่งนี้จะส่งผลให้ระบบปิดบริการต่างๆ ถอนการเชื่อมต่อโครงสร้างไฟล์ของระบบ และหยุดการทำงานของทุกโพรเซสที่ทำงาน
Syntax	**reboot [*options*]**
Example	**# reboot**

options:

-d เริ่มต้นทำงานใหม่ และไม่บันทึกข้อมูลในไฟล์ /var/log/wtmp ซึ่งเป็นไฟล์ที่ใช้บันทึกข้อมูลการ login และ logout

-f สั่งให้ระบบทำการ restart ทันทีโดยไม่มีเงื่อนไข (force)

-i สั่งให้การ์ดเน็ตเวิร์คหยุดการทำงานก่อน restart เครื่องใหม่

-n สั่งเริ่มการทำงานของเครื่องใหม่โดยไม่ต้องเรียกไปยัง sync (sync คือโพรเซสที่ทำหน้าที่บันทึกข้อมูลจากหน่วยความจำลงฮาร์ดดิสก์)

-w สั่งเริ่มการทำงานของเครื่องใหม่โดยบันทึกข้อมูลการ login, logout ลงในไฟล์ /var/log/wtmp

ตัวอย่างการใช้งาน :

reboot สั่ง shutdown ระบบและปิดเครื่อง

ลองทดสอบ :

```
# reboot -s
# reboot -w
```

ดูคำสั่งอื่นประกอบ : *shutdown, init, /sbin/poweroff, /sbin/halt*

คำสั่ง shutdown

System Management : **shutdown**	
คำสั่ง	**shutdown** เป็นคำสั่งที่ให้ระบบปฏิบัติการเริ่มต้นการทำงานใหม่ หรือปิดระบบ
Syntax	***shutdown [-akrhHPfnc] [-t secs] time [warning message]*** ***/sbin/shutdown***
Example	**# shutdown –h now** **# shutdown +5 "Server is going DOWN about 5 minutes !!!"**

options :

-a สั่งปิดระบบโดยมีเงื่อนไงคือผู้ที่จะปิดระบบได้ต้องมีรายชื่ออยู่ในไฟล์ /etc/shutdown.allow (ในกรณีที่มีการเรียกมาจาก init)

-k ไม่มีการปิดระบบใดๆ เพียงแต่แจ้งเตือนไปให้ทุกๆ user ที่ผู้ในระบบทราบ

-r สั่งให้ปิดระบบจากนั้นให้ทำการเริ่มต้นใหม่ (reboot after shutdown)

-h สั่งปิดระบบและหยุดการทำงานของเครื่องด้วย

-P สั่งปิดระบบแต่ให้ผู้ใช้งานกดปุ่ม power ของเครื่องเอง

-H สั่งปิดระบบและเข้าสู่โหมดการ monitor ของเครื่อง(ฮาร์ดแวร์ต้องสนับสนุนแบบ boot monitor)

-f สั่งปิดระบบ และไม่ต้องทำการตรวจสอบโครงสร้างไฟล์(fsck)

-F สั่งปิดระบบ และต้องทำการตรวจสอบโครงสร้างไฟล์ด้วย(fsck)

-c ยกเลิกกระบวนการ shutdown ที่กำลังดำเนินการอยู่

-t secs ระยะเวลาที่รอให้ระบบทำการแจ้งเตือนผู้ใช้และส่งสัญญาณหยุดโพรเซสที่กำลังทำงานอยู่

time เวลาที่ต้องการจะปิดระบบ

warning-message ข้อความที่ต้องการส่งไปให้ผู้ใช้งานก่อนทำการปิดระบบ

ตัวอย่างการใช้งาน :

shutdown –h now

สั่ง shutdown ระบบและปิดเครื่องด้วย ทันที

shutdown now "Server going down now!!!"

สั่ง shutdown ระบบทันที พร้อมแสดงข้อความ Server goning down now!!!

shutdown -r +10 "message"

เหลือเวลาอีก 10 นาที จะ restart เครื่อง พร้อมแสดงข้อความ message

shutdown 8:00

```
สั่ง shutdown ระบบ ที่เวลา 8.00 นาฬิกา
root@localhost ~]# shutdown –r now
สั่งเริ่มต้นระบบใหม่ ทันที
# shutdown -rF now
สั่ง restart เครื่องใหม่ทันทีพร้อมกับตรวจสอบโครงสร้างไฟล์ด้วย
# shutdown +5 "*** Server is going DOWN for hard disk replacement!!!
Please save all your work ***"
สั่งให้อีก 5 นาที จะทำการปิดระบบ พร้อมแสดงข้อความ "Server is going …." ด้วย
# shutdown 18:00 "SERVER DOWN"
สั่งปิดระบบ ที่เวลา 18.00 น และแสดงข้อความว่า SERVER DOWN
# shutdown -rF now
สั่ง restart เครื่องใหม่ทันทีพร้อมกับตรวจสอบโครงสร้างไฟล์ด้วย
```

ลองทดสอบ :

```
# shutdown -h 0
# su -c "/sbin/shutdown -h now"
# telinit 0
```

ดูคำสั่งอื่นประกอบ : *halt, poweroff, reboot*

5.3.20 กลุ่มคำสั่งเกี่ยวกับ Shortcuts

คำสั่ง CTRL + C เป็นคำสั่งที่ใช้งานในเชลล์ได้ทุกเชลล์ เพื่อยกเลิกคำสั่งที่กำลังทำงานอยู่ในปัจจุบัน

คำสั่ง CTRL + Z ใช้สำหรับคำสั่งที่ใช้งานปัจจุบัน

คำสั่ง CTRL + D เป็นคำสั่งที่ใช้สำหรับส่งสัญญาณบอกให้โปรแกรมที่กำลังทำงานทราบว่าข้อมูลสิ้นสุดลงแล้ว หรือ logout ออกจากเชลล์

คำสั่ง Ctrl+W เป็นคำสั่งสำหรับยกเลิกข้อความที่พิมพ์มาแล้ว 1 คำ

คำสั่ง !! ใช้สำรับเรียกคำสั่งล่าสุดที่ทำงานแล้วกลับมาทำงานใหม่อีกครั้ง

คำสั่ง exit ใช้สำหรับออกจากโปรแกรมที่กำลังทำงานอยู่หรือออกจากเชลล์ที่ใช้งาน

5.3.21 กลุ่มคำสั่งเกี่ยวกับ Compiler

ในการพัฒนาโปรแกรมสิ่งที่ขาดไม่ได้เลยก็คือ Tools ในการพัฒนา เช่น Compiler และ Utility ต่างๆ ใน Unix นั้นมีอยู่ 2 โปรแกรมยอดนิยมที่นักพัฒนาโปรแกรมรู้จักกันดีก็คือ gcc และ make

คำสั่ง gcc

gcc ย่อมาจาก GNU C Compiler เป็น Compiler ที่แปลภาษา C ให้เป็นภาษาเครื่อง (Machine Language) และทำการ Link เพื่อให้เกิด File ที่สามารถนำมาทำงานได้ (Executable File) การทำงานของ gcc มีดังต่อไปนี้

- Preprocessing เป็นกระบวนการที่รับ input ที่เป็นภาษา C (Source Code) เข้ามาทำงานในส่วนของ Preprocessor Directives (บรรทัดที่ขึ้นต้นด้วย # เช่น #include, #define) และตัดส่วนของ comment ออกจากโปรแกรม
- Compilation เป็นกระบวนการแปลภาษา C (Source Code) ให้เป็นภาษา Assembly (Assembly Code)
- Assembly เป็นกระบวนการแปลงภาษา Assembly Code ให้เป็น Object Code ซึ่งจะเป็นภาษาเครื่องแต่ยังไม่สามารถ execute ได้เนื่องจากยังไม่ได้ทำการ Link กับ Library
- Linking เป็นกระบวนการ Link Object Code เข้ากับ Library ซึ่งจะได้ output file ที่เป็น executable file

การใช้งาน gcc มีรูปแบบดังนี้

gcc [option | filename]

โดย option ที่ควรทราบและมักจะต้องใช้งานบ่อยๆ มีดังนี้

-c เป็นการ Compile source code ให้เป็น object code โดยไม่ทำการ link ผลลัพธ์ที่ถูก compile จะเป็นชื่อ file ของ source code แต่จะมี extention เป็น .o เช่น หาก Source File ชื่อ main.c ใช้คำสั่ง gcc -c main.c จะได้ผลลัพธ์เป็น object code ซึ่งจะเป็น file ชื่อ main.o

-o file เป็นการกำหนดชื่อ output file ให้เป็นชื่อ file ที่ระบุหลัง option -o ในกรณีที่ไม่ใช้ option -c ร่วมด้วยผลลัพธ์ที่ได้จะเป็น executable file เช่น คำสั่ง gcc -o main main.c คือการสั่งแปล source code ภาษา C ชื่อ main.c ให้มีผลลัพธ์เป็น executable file ชื่อ main แต่ในกรณีที่ไม่ระบุชื่อ output file ผลลัพธ์ที่ได้ เป็น executable file ที่ชื่อ a.out เช่น gcc main.c จะได้ผลลัพธ์เป็น file a.out

ตัวอย่างการใช้งานจริงในการ compile โปรแกรมที่มีหลาย module

gcc -c main.c -o objfile1 (compile main program ให้เป็น object file)

gcc -c module1.c -o objfile2 (compile module1 ให้เป็น object file)

gcc -c module2.c -o objfile3 (compile module2 ให้เป็น object file)

gcc objfile1 objfile2 objfile3 -o executablefile (link object file ของ main program และ module เข้าด้วยกันให้เป็น executablefile) File Type ที่ควรทราบก่อนในการพัฒนาโปรแกรมด้วย gcc มีดังนี้

- .c เป็น C source code
- .h เป็น C header file
- .o เป็น object file
- a.out เป็น output ที่เป็น executable file เมื่อ compile แบบไม่ระบุชื่อ file output

คำสั่ง make

จากตัวอย่างของการ compile ที่มีหลายๆ module เราจะเห็นว่า ถ้ามีการเปลี่ยนแปลง source code ของ module ใด เราจะต้อง compile module นั้นๆ ให้เป็น object file ใหม่ จากนั้นก็นำ object ตัวใหม่ไป link เข้ากับ object file ตัวเก่า เพื่อให้ได้ executable file เช่น ถ้าเราต้องการแก้ไข source ใน module2 ขั้นตอนการ compile ใหม่มีดังนี้

```
gcc -c module2.c -o objfile3
gcc objfile1 objfile2 objfile3 -o executablefile
```

จะเห็นได้ว่า ถ้าเรายิ่งมี module file เยอะ การ compile ก็จะยิ่งไม่สะดวกตามไปด้วย make เป็นโปรแกรม Utility ซึ่งใช้ในการจัดการกลุ่มของคำสั่ง และสามารถตรวจสอบว่า file ใดมีความจำเป็นจะต้องทำการ update บ้าง เช่น ทำการ compile ใหม่ link ใหม่ ทำการเชื่อมโยง กับโปรแกรมหรือไฟล์ใดบ้าง โดยการที่จะให้โปรแกรม make ทำสิ่งดังที่กล่าวมาได้ เราจะต้องกำหนด ความสัมพันธ์ต่างใน makefile เสียก่อน ดังนั้นโปรแกรม make จึงเป็นโปรแกรมที่มีความสามารถใช้ในการจัดการกับโปรแกรมในภาษาใดก็ได้ หรือ งานประเภทใดก็ได้ที่มีลักษณะขึ้นต่อกัน

ลักษณะของ makefile

การสร้าง makefile ทำได้โดยสร้างเป็น textfile ธรรมดา เพียงแต่ให้ตั้งชื่อ file ว่า "makefile" การเขียนความสัมพันธ์ระหว่างไฟล์ต่างๆ ใน makefile มีตัวอย่างดังนี้ (ยกตัวอย่างตามตัวอย่างการ compile ที่มีหลาย module ด้านบน)

```
main: main.o module1.o module2.o
gcc -o main main.o module1.o module2.o
main.o: main.c
gcc -c main.c -o main.o
module1.o: module1.c
gcc -c module1.c -o module1.o
module1.o: module2.c
gcc -c module2.c -o module2.o
clean:
rm -rf *.o
```

บทสรุป

คำสั่งบนระบบปฏิบัติการยูนิกซ์และลินุกซ์มีมากมายหลายประเภท ซึ่งแบ่งออกเป็นกลุ่มๆ ได้มากมาย ในเอกสารฉบับนี้แบ่งออกทั้งหมด 21 กลุ่ม แต่ละกลุ่มเรียงตามอักษร การเข้าใจการทำงานของแต่ละคำสั่ง จำเป็นต้องมีการปฏิบัติและทดลองด้วยตนเอง

คำถามท้ายบท

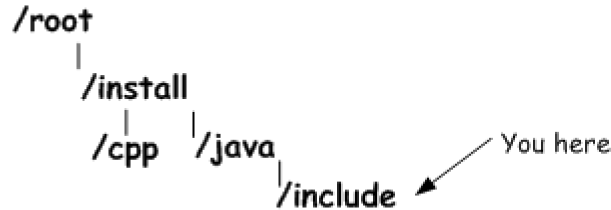
1. จงแสดงสถาปัตยกรรมของเครื่องที่กำลังใช้งานอยู่
2. จงแสดงปฏิทินของ เดือนนี้ ในปีปัจจุบัน
3. จงแสดงปฏิทินของ เดือน เมษายน ในปี 2009
4. จงแสดงปฏิทินของปี 2012
5. จงแสดงปฏิทินของ เดือนที่ผ่านมา เดือนปัจจุบัน และเดือนหน้า โดยให้วันจันทร์เป็นวันแรกของปฏิทิน
6. จงแสดงรายละเอียดของซีพียูที่ใช้งาน อินเทอร์รัพท์ หน่วยความจำ swap เวอร์ชันของเคอร์เนล ข้อมูลการใช้งานของการ์ดเน็ตเวิร์ค รายละเอียดการ mount ไฟล์
7. จงแสดงเวลาของระบบด้วยคำสั่ง clock
8. จงแสดงวัน เวลาของระบบให้มีรูปแบบดังต่อไปนี้

1	DATE: วันที่ปัจจุบัน TIME:เวลาปัจจุบัน
2	Hello Date is วันที่ปัจจุบัน Time is เวลาปัจจุบัน

9. จงกำหนดวัน เวลา ของระบบเป็น วันเสาร์ที่ 19 เดือนเมษายน ค.ศ. 2007 เวลา 9:35:10 น
10. จงแสดงข้อมูลของของ BIOS ในส่วนของ System ด้วยคำสั่ง dimdecode
11. จงแสดงข้อมูลของฮาร์ดดิสก์ที่ใช้งาน
12. จงแสดงข้อมูลของการ์ด pci ในระบบ
13. จงแสดงข้อมูลของ USB ในระบบ
14. จงแสดงข้อมูลพื้นฐานของระบบที่ใช้งาน เช่น ชื่อเครื่อง เวอร์ชัน สถาปัตยกรรม เป็นต้น
15. สั่งให้ระบบทำการเริ่มต้นทำงานใหม่ ด้วย init
16. สั่งให้ระบบเริ่มทำงานใหม่และเข้าสู่การทำงานแบบ เข้าใช้งานได้หลายคน และควบคุมการทำงานผ่านทาง command line เท่านั้น
17. คำสั่งออกจากเชลล์ที่กำลังใช้งานอยู่
18. คำสั่งที่ทำให้ระบบปฏิบัติการยูนิกส์และลินุกซ์ เริ่มต้นทำงานใหม่คืออะไร
19. จงสั่งให้ปิดระบบอีก 15 นาที โดยมีข้อความเตือนว่า “Server is going down !!!”
20. จงสั่งให้ปิดระบบพร้อมปิดเครื่องด้วยทันที
21. จงสั่งให้ระบบทำการ restart ใหม่ภายในเวลา 3 นาทีที่จะมาถึง
22. จงย้ายไดเรคทรอรีไปยัง /var/log

23. จากรูป ผู้ใช้งานอยู่ที่ไดเรคทรอรี include ถ้าออกคำสั่งต่อไปนี้จะเกิดอะไรขึ้น

    cd ../install/cpp



24. จงย้ายไปยัง home ไดเรคทรอรีของผู้ใช้ ด้วยการใช้คำสั่งแบบ relative path
25. จงแสดงไดเรคทรอรีและไฟล์ใน /etc/init.d/ แบบละเอียดคือ แสดงไฟล์ซ่อน ., .. เจ้าของไฟล์ ขนาดไฟล์ วันเวลา และชื่อไฟล์
26. จงแสดงไฟล์ที่สามารถประมวลผลได้ในไดเรคทรอรี่ /usr/bin
27. จงแสดงไฟล์ที่ถูกใช้งานล่าสุดมาก่อนในไดเรคทรอรี /etc
28. ให้ทำการสำรองไฟล์ที่มีชื่อว่า /root/insatll.log ไปไว้ในไดเรคทรอรี /home
29. ทำการสำรองไดเรคทรอรีชื่อว่า /var/log/ ไปยัง /usr/local เป็นชื่อใหม่คือ bklog
30. ทำการสำรองข้อมูลทุกไฟล์ที่มีส่วนขยายเป็น .txt
31. ค้นหาไฟล์ที่ขึ้นต้นด้วยคำว่า inst
32. ค้นหาไฟล์ที่มีชื่อขึ้นต้นและลงท้ายด้วยอักษรอะไรก็ได้ กี่ตัวก็ได้ แต่ตรงกลางต้องมีคำว่า all
33. ค้นหาไฟล์ที่มีชื่อขึ้นต้นและลงท้ายด้วยอักษรอะไรก็ได้ กี่ตัวก็ได้ แต่ตรงกลางต้องมีอักษรว่า Doc เท่านั้น ค้นหาไฟล์ในไดเรคทรอรี /usr/ เมื่อค้นหาเจอแล้วให้ทำคำสั่งลบไฟล์ที่หาเจอนั้นทิ้งให้หมด
34. ค้นหาไฟล์ที่ถูกแก้ไขต่ำกว่า 2 วัน
35. ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรี /home เลือกเฉพาะไฟล์ของผู้ใช้ชื่อ test เท่านั้น
36. ค้นหาไฟล์ทุกไฟล์ ภายใต้ไดเรคทรอรีปัจจุบัน ซึ่งไฟล์ที่ค้นหามีเงื่อนไขคือ เจ้าของอ่านและเขียนได้ สมาชิกในกลุ่มสามารถอ่านได้ กลุ่มอื่นๆอ่านได้อย่างเดียว
37. ค้นหาไฟล์ใดๆ ก็ได้ที่มีขนาดไฟล์มากกว่า 15 MB
38. ค้นหาไฟล์จาก /usr/ ไฟล์ที่ลงท้ายด้วย .java ที่แก้ไขมาแล้วไม่เกิน 5 นาที
39. ค้นหาไฟล์ที่ไม่ได้ขึ้นต้นด้วยอักษร A-Z ที่เป็นตัวใหญ่ เมื่อพบพิมพ์ใส่ไฟล์ชื่อ foo.txt
40. ต้องการพิมพ์ไฟล์ชื่อที่ขึ้นต้นด้วย t* ด้วย option –regex ในไดเรคทรอรีปัจจุบัน
41. สร้างลิงค์เชื่อมโยงจากไฟล์ชื่อ sshd ใน /etc/init.d/ ให้เป็นชื่อ lnsshd พร้อมแสดงสัญลักษณ์เชื่อมโยงลิงค์ให้เห็น
42. ให้ทำการสร้างไดเรคทรอรีชื่อว่า work ในไดเรคทรอรีปัจจุบัน จากนั้นสร้างไดเรคทรอรีย่อยไว้ภายในอีก คือ mgr, net, acc ภายใน net ให้สร้างอีก 2 ไดเรคทรอรีคือ it และ cs

43. สร้างไดเรคอทรอรีชื่อว่า person และมีสิทธิ์ในการเข้าถึงไฟล์ คือเจ้าของ สมาชิกในกลุ่ม และบุคคลอื่นๆ สามารถเขียนไฟล์นี้ได้อย่างเดียวเท่านั้น
44. ให้สร้างไดเรคทรอรีดังต่อไปนี้ /work/it/network/os โดยใช้ option –p
45. ย้ายแฟ้มทุกแฟ้มไปไว้ในไดเรคทรอรี /tmp
46. เปลี่ยนชื่อไฟล์จาก pic1.jpg เป็น pic2.jpg
47. เปลี่ยนชื่อไดเรคทรอรีชื่อ old_dir เป็น new_dir
48. เปลี่ยนชื่อไฟล์จาก file1 เป็นไฟล์ file2 แต่ต้องให้ยืนยันก่อนทำการเปลี่ยนชื่อ
49. จงแสดง path ของไดเรคทรอรีปัจจุบันที่กำลังทำงานอยู่
50. ลบโฟล์เดอร์ test และหากมีโฟล์เดอร์หรือไฟล์อะไรอยู่ใน /test ก็ลบทั้งหมดทิ้งด้วย
51. ลบไฟล์ชื่อว่า myfile.txt โดยต้องมีการยืนยันการลบด้วย
52. ลบไฟล์ชื่อว่า file1 file2 file3 พร้อมๆ กับมีการยืนยันการลบทุกครั้ง
53. สร้างไฟล์ใหม่ที่ว่างเปล่า 2 ไฟล์ชื่อว่า test1 test2
54. เปลี่ยนค่าเวลา access time และเปลี่ยนเวลา modification time ของ test1 เป็นเวลาปัจจุบัน
55. เปลี่ยนเวลาของ test2 เป็น 1 Jan 2010 11:30
56. เปลี่ยนเวลาของ test1 เป็น ปี 2015 เดือน 9 วันที่ 14 เวลา 8.10 นาที 50 วินาที
57. แสดงโครงสร้างของไฟล์และไดเรคทรอรีทั้งหมดในรูปแบบโครงสร้างทรี
58. แสดงโครงสร้างทรีลึกแค่ 3 ลำดับเท่านั้น
59. แสดงโครงสร้างทรีเฉพาะไดเรคทรอรี ใน /etc เท่านั้น และทรีลึกเพียงแค่ 3 ระดับ และสัญลักษณ์เส้นของทรีเป็นสีแบบ ASCII code และเป็นเส้นทึบ
60. จงหาตำแหน่งที่ติดตั้งของโปรแกรม perl ที่ติดตั้งในระบบด้วยคำสั่ง locate
61. ค้นหาตำแหน่งของไดเรคทรอรีชื่อ InsTALl
62. ค้นหาไบนารี ไฟล์ต้นฉบับและคู่มือของ find
63. ค้นหาตำแหน่งของคำสั่ง ssh
64. สร้างบัญชีผู้ใช้ใหม่ชื่อว่า Harry เป็นสมาชิกกลุ่ม users มีหมายเลข UID คือ 1005, home ไดเรคทรอรีคือ /home/Harry มี comment คือ Harry Potter
65. จงลบรายชื่อผู้ใช้ชื่อ somsri ออกจากระบบ โดยไม่ต้องลบ Home ไดเรคทรอรีออกด้วย
66. จงลบรายชื่อผู้ใช้ชื่อ somsri ออกจากระบบ พร้อมกับข้อมูลของผู้ใช้ดังกล่าวออกทั้งหมด
67. สร้างรายชื่อกลุ่มใหม่ชื่อว่า NGROUP ในระบบ
68. สร้างรายชื่อกลุ่มใหม่ชื่อว่า IT_GROUP ในระบบโดยมี GID = 550
69. เปลี่ยนหมายเลข GID ใหม่ สมมุติว่าเดิม GID ของ ENG เป็น 450 ให้เปลี่ยนเป็น 451
70. จากข้อมูลของกลุ่มในไฟล์ /etc/group ดังนี้

    ENG:x:**451**:

IT:x:**500**:

ให้เปลี่ยน GID ของ IT จากเดิมคือ 500 เป็น 451 เหมือนกับกลุ่ม ENG

71. ให้ทำการตรวจสอบความถูกต้องของข้อมูลในไฟล์ gshadow พร้อมแก้ไขให้ถูกต้อง
72. สั่งให้ทำการเปลี่ยนใส่รหัสผ่านของผู้ใช้ชื่อ tom ใหม่
73. ลบรหัสผ่านของผู้ใช้งานชื่อ tom ออก
74. กำหนดเวลาการใช้งานของผู้ใช้ tom สูงสุดไม่เกิน 90 วัน
75. กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่าน เขียนได้ สมาชิกภายในกลุ่มอ่าน เขียนได้ และกลุ่มอื่นๆ สามารถอ่าน เขียนได้ บนไฟล์ชื่อ chfile.txt
76. กำหนดสิทธิ์ให้เจ้าของไฟล์สามารถอ่าน เขียน และประมวลผลได้ สมาชิกภายในกลุ่มสามารถอ่านและประมวลผลได้ และกลุ่มอื่นๆ ประมวลผลได้ บนไฟล์ชื่อ file
77. กำหนดสิทธิ์ให้เจ้าของไฟล์ อ่านและประมวลผลได้ สมาชิกในกลุ่มและกลุ่มอื่นๆ อ่านได้อย่างเดียว ถ้าไดเรคทรอรี documents มีไดเรคทรอรีย่อยก็ให้กำหนดสิทธิ์ลงไปทุกๆ ไดเรคทรอรีย่อยทั้งหมดด้วย
78. จงเปลี่ยนสิทธิ์ไฟล์ชื่อ data.txt มีเจ้าของใหม่เป็นชื่อ sutida
79. เปลี่ยนสิทธิ์ให้ไดเรคทรอรีชื่อ install มีเจ้าของใหม่เป็นชื่อ sutida ถ้าไดเรคทรอรี install มีไดเรคทรอรีย่อยภายใน ให้รับสิทธิ์ในการเปลี่ยนไปด้วย
80. เปลี่ยนสิทธิ์ให้ไดเรคทรอรีและไดเรคทรอรีย่อย ชื่อ documents มีเจ้าของใหม่เป็นชื่อ test เป็นสมาชิกในกลุ่ม users
81. จงใช้คำสั่งป้องกันการลบไฟล์ /log/test_log จาก root หรือโพรเซส หรือผู้ใช้ แต่ยังสามารถเขียนข้อมูลเพิ่มเข้าไปได้ พร้อมทั้งแสดงผลการป้องกัน
82. ป้องกันการลบหรือการแก้ไขไฟล์ test.conf จาก root หรือโพรเซส หรือผู้ใช้อื่นๆ พร้อมทั้งแสดงผลการป้องกัน
83. ทำการสำรองไฟล์ข้อมูลทุกๆ ไฟล์ที่อยู่ในไดเรคทรอรี home ไปไว้ยัง /backup/ ชื่อว่า backup01
84. ทำการกู้คืนไฟล์ในข้อ 83 กลับคืนมา
85. ทำการบีบอัดข้อมูลก่อนสำรองไฟล์ข้อมูล อยู่ใน /home/ myfile.txt ไปเก็บไว้ใน /backup/ ชื่อว่า myfile.tar.gz
86. ไดเรคทรอรีปัจจุบันมีไฟล์ทั้งหมด 5 ไฟล์คือ a.txt, b.txt, c.txt, d.txt, e.txt ให้ทำการสำรองข้อมูลชื่อว่า BK1.tbz แต่เลือกสำรองเฉพาะไฟล์ชื่อว่า a.txt, d.txt และ e.txt เท่านั้น
87. ทำการบีบอัดไฟล์ 2 ไฟล์พร้อมกัน(ด้วย bzip2) พร้อมกับลบไฟล์ต้นฉบับ
88. คลายการบีบอัดไฟล์(ด้วย bunzip2)
89. ทำการบีบอัดข้อมูลด้วยคำสั่ง tar ร่วมกับ bzip2 พร้อมกันในครั้งเดียว

90. บีบอัดข้อมูลไฟล์ชื่อ myfile ไฟล์ที่ถูกบีบอัดแล้วจะมีส่วนขยายเป็น .gz
91. บีบอัดข้อมูลทุกไฟล์ในไดเรคทรอรี dir1 แต่ไม่บีบอัดไดเรคทรอรี dir1
92. แสดงข้อมูลที่อยู่ในไฟล์รหัสผ่าน ด้วยคำสั่ง cat
93. แสดงข้อมูลที่อยู่ในไฟล์ file1.txt โดยนับจำนวนบันทัด รวมบันทัดว่าง
94. เชื่อมต่อข้อมูล (append) ไปยังไฟล์ file1.txt
95. ดูแฟ้มข้อมูลทีละ 1 หน้าจอ ไฟล์ชื่อว่า /var/log/messages
96. เคลียร์หน้าจอก่อนแสดงผลข้อมูลในไฟล์ /var/log/messages
97. แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ file1.txt
98. แสดงข้อมูล 20 บันทัดแรกในส่วนหัวของไฟล์ชื่อ file1.tx
99. แสดงข้อมูลในส่วนหัวของไฟล์ชื่อ file1.txt ไม่เกิน 1 กิโลไบต์ไบต์
100. แสดงข้อมูลในไฟล์ file.txt ครั้งละ 1 จอภาพ โดยสามารถเลื่อนดูไฟล์ไปยังไฟล์และท้ายไฟล์ได้
101. แสดงข้อมูลในส่วนท้ายของไฟล์ 10 บันทัด
102. แสดงข้อมูลในส่วนท้ายของไฟล์ จำนวน 20 บันทัด
103. แสดงข้อมูลไฟล์ที่มีขนาดใหญ่มาก โดยแสดงแค่ 20 ไบต์เท่านั้น
104. แสดงข้อมูลในส่วนท้ายของไฟล์ message.log อย่างต่อเนื่องแบบ realtime
105. แสดงข้อความ hello world ออกจอภาพ
106. แสดงข้อความ Warning !!! ออกจะภาพ พร้อมกับเสียง bell
107. ให้แสดงเมนูดังต่อไปนี้ทางจอภาพ

```
**********************************
*             MAIN MENU          *
**********************************
*      1. Computer Programming   *
*      2. Database Design        *
*      3. Operating System       *
*      4. Exit                   *
**********************************
```

108. จงเปรียบเทียบไฟล์ 2 ไฟล์ที่ได้รับการจัดเรียงลำดับมาแล้ว
109. เปรียบเทียบความแตกต่างระหว่างระหว่าง myfile1 และ myfile2 โดยไม่สนใจข้อความว่างที่อยู่ในไฟล์ทั้งหมด
110. ต้องการหาคำว่า mail ในไฟล์ /etc/passwd ว่ามีอยู่ในบรรทัดใดบ้าง ผลลัพธ์ที่ได้จะแสดงบรรทัดที่มีคำว่า mail อยู่ด้วย

111. ค้นหาคำว่า mail ในไฟล์ /etc/passwd ผลลัพธ์จะมีเลขที่บรรทัดกำกับมาด้วย

112. ต้องการนับจำนวนคำว่า nologin ที่พบในไฟล์ /etc/passwd

113. ต้องการค้นหาคำว่า Connect:127.0.0.1 ในไดเรคทรอรี /etc/ รวมถึงไดเรคทรอรีย่อยทั้งหมดที่อยู่ใน /etc ด้วย

114. ต้องการค้นหาข้อความ cpu ในไฟล์ /proc/cpuinfo

115. แสดงสีของข้อความที่ต้องการค้นหา คือ tty ในไฟล์ /etc/group

116. แสดงข้อมูลที่ได้จาก ls เป็น 3 คอลัมภ์

117.

จากตัวอย่างข้อมูล

f1.txt	f2.txt
Apple Banana Mango	30$ 25$ 45$

จงสั่งแสดงผลลัพธ์ดังต่อไปนี้

```
Apple=30$
Banana=25$
Mango=45$

Apple+Banana+Mango
30$+25$+45
```

118.

ตัวอย่าง file1.txt

```
hscripts has many valuable free scripts
It is the parent site of www.forums.hscripts.com
hscripts include free tutorials and free gif images
free DNS lookup tool
Purchase scripts from us
A webmaster/web master resource website
```

ทำการค้นหาคำว่า DNS ใน file1.txt เมื่อคนหาเจอจะแทนที่ด้วย คำว่า Domain Name Server

119. ลบเฉพาะข้อความที่มีชื่อ tool ในไฟล์ file1.txt ออกเท่านั้นโดยไม่แก้ไขข้อความอื่นๆทั้งสิ้น

120. แสดงผลข้อมูลในไฟล์ file1.txt เฉพาะบันทัดที่มีข้อความว่า hscripts เท่านั้น

121. จัดเรียงข้อมูลตามลำดับตัวอักษรใน file.txt

122. จัดเรียงข้อมูลที่ได้จากคำสั่ง ls –al โดยเรียงตามขนาดของไฟล์ (เรียงจากมากไปน้อย)

123. แสดงข้อมูลในไฟล์ myfile.txt โดยมีหมายเลขบรรทัดระบุ ในส่วนต้นของบรรทัด

124. แสดงข้อมูลในไฟล์ myfile.txt เป็นเลขฐานแปด

125. แสดงข้อมูลในเชลล์ sh โดยกำหนดรูปแบบการแสดงผลเป็นเป็นเลขฐานสิบหก
126. จงรายงานคุณสมบัติของไฟล์
127. นับจำนวนแถว คำ และตัวอักษร ในแฟ้มข้อมูล file1.txt
128. นับเฉพาะจำนวนบรรทัดและตัวอักษรในแฟ้ม /etc/passwd
129. แสดงเนื้อที่การใช้งานของไดเรคทรอรี /etc
130. แสดงเนื้อที่การใช้งานของไดเรคทรอรี /etc แบบสรุป(แสดงเฉพาะขนาดรวมทั้งหมด)
131. จงตรวจสอบชนิดของไฟล์ในไดเรคทรอรีปัจจุบัน
132. ค้นหาตำแหน่งของคำสั่ง ls โดยแสดงผลเฉพาะไฟล์ที่ประมวลผลได้เท่านั้น
133. ค้นหาตำแหน่งของคำสั่ง ls โดยแสดงตำแหน่งที่เก็บไฟล์ประมวลผล และคู่มือ
134. คัดแยกคำจากไฟล์ชื่อ /etc/passwd เลือกเฉพาะ user name, userid และตำแหน่งของไดเรคทรอรี Home
135. คัดแยกคำจากไฟล์ชื่อว่า /etc/passwd โดยเลือกเอาเฉพาะอักษรตำแหน่งที่ 1- 5
136. คัดแยกคำจากไฟล์ชื่อว่า /etc/passwd โดยเลือกเอาเฉพาะอักษรตำแหน่งที่ 1-5 และ 7-15 โดยใช้เพียง 1 คำสั่งเท่านั้น
137. นำข้อมูลของแต่ละแถวในแฟ้ม file1 และ file2 มาเชื่อมต่อกันแล้วเก็บไว้ใน file3
138. ทำการเชื่อมต่อแฟ้มข้อมูล file1 และ file2 เข้าด้วยกัน แบบต่อเนื่องเป็นแถวเดียวโดยใช้อักษร delimiter คือ *
139. แสดงรายการในไดเรคทรอรีปัจจุบันให้มีคอลัมภ์เท่ากับ 3 คอลัมภ์
140. จงแปลงอักษรในแฟ้มข้อมูลเป็นตัวอักษรใหญ่ทั้งหมด
141. แสงผลข้อความที่ซ้ำกันในแฟ้มข้อมูล ให้เหลือเพียงบรรทัดเดียวเท่านั้น
142. ส่งผลลัพธ์จากคำสั่ง date ไปเก็บไว้ในแฟ้ม file1และส่งไปยังคำสั่ง less ด้วย
143. จงเปรียบเทียบความแตกต่างระหว่างไฟล์ file1 และ file2
144. จงเปรียบเทียบความแตกต่างระหว่างไฟล์ file1 และ file2 เมื่อไฟล์เหมือนกันให้พิมพ์ “no diff” แต่ถ้าแตกต่างกันไม่ต้องแสดงข้อความใดๆ
145. ทำการสร้างรหัส md5 ของไฟล์ /etc/passwd และส่งไปเก็บไว้ยังไฟล์ชื่อว่า myPasswd.md5
146. ทำการตรวจสอบความถูกต้องของไฟล์ไฟล์ /etc/passwd ด้วย myPassed.md5
147. แสดงรายงานการใช้พื้นที่ดิสก์ พื้นที่เหลือ และพาร์ติชัน
148. รายงานการใช้ดิสก์ โดยแสดงไฟล์ทั้งหมดที่ใช้ในระบบ แสดงผลแบบอ่านง่าย และพิมพ์ชื่อชนิดของไฟล์ด้วย
149. ทำการ mount ไดเรคทรอรี /var/log เข้ากับโครงสร้างไฟล์ใหม่ที่สร้างขึ้นคือ /usr/logs
150. ทำการ mount อุปกรณ์ floppy disk เข้ากับโครงสร้างไฟล์ชื่อว่า /mnt /floppy

151. ทำการ mount อุปกรณ์ flash drive เข้ากับโครงสร้างไฟล์ชื่อว่า /mnt/flash
152. ทำการ mount อุปกรณ์ cd-rom drive
153. สั่งให้ทำการตรวจสอบความผิดปกติของไฟล์ทั้งหมดในระบบ
154. สั่งให้ทำการตรวจสอบความผิดปกติของข้อมูลใน /dev/hd1(ฮาร์ดดิกส์ชนิด IDE) หรือดิสก์ที่ติดตั้งอยู่ในระบบตัวใดตัวหนึ่ง
155. สั่งสำรองข้อมูลด้วยคำสั่ง dump ไดเรคทรอรีชื่อว่า /var/log เป็นชื่อว่า bklog
156. สั่งกู้คืนข้อมูลด้วยคำสั่ง restore จาก dump-file ชื่อ bklog ในข้อที่ 155 กลับคืนมา
157. สั่งสำรองข้อมูลจากเครื่องโฮสต์ sync server ไปยังเครื่องผู้ใช้
158. สั่งสำรองข้อมูลกลับจากเครื่องผู้ใช้กับเครื่อง sync server
159. สั่งพิมพ์ข้อมูลในไฟล์ /etc/passwd ออกทางเครื่องพิมพ์
160. แสดงโพรเซสที่กำลังทำงานเต็มรูปแบบ
161. แสดงข้อมูลของ process พร้อมชื่อโปรแกรม ชื่อผู้สั่ง ซีพียู หน่วยความจำ และข้อมูลอื่นๆ อย่างละเอียด
162. แสดงโพรเซสที่กำลังทำงานด้วยคำสั่ง w
163. แสดงการทำงานของโพรเซสที่มีอยู่ในระบบทั้งหมด
164. แสดงการทำงานของโพรเซสเฉพาะของผู้ใช้งานชื่อ httpd
165. รายงานการใช้หน่วยความจำที่กำลังใช้งาน ซึ่งจะ update ทุกๆ 5 วินาที โดยรายงานข้อมูลมีหน่วยวัดเป็นกิโลไบต์
166. จงยกเลิกโพรเซสใดโพรเซสหนึ่งในระบบ โดยใช้ออพชัน SIGTERM, SIGHUP
167. จงอธิบายความหมายของ signal ในคำสั่ง kill มาอย่างน้อย 5 signal
168. จงเพิ่มลำดับความสำคัญของโพรเซสให้มีค่าเท่ากับ 5, 7, 12 ตามลำดับ
169. จงแสดงหน่วยความจำที่ใช้งาน (free) ทุกๆ 3 วินาที
170. จงตรวจสอบการทำงานของการ์ดเน็ตเวิร์ค(eth0) ทุกๆ 1 วินาที พร้อมทั้ง highlight ข้อมูลที่มีการเปลี่ยนแปลงด้วย
171. จงสั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh รันในวันที่ 25 เดือนเมษายน 2010 เวลา 00:00 นาที
172. จงสั่งให้รันเชลล์สคริปต์ชื่อว่า myjob.sh อีก 3 เดือนข้างหน้า เวลา 18:00 นาที
173. สั่งให้คำสั่ง ls –l ทำงานทุกๆ 5 นาที
174. สั่งให้ทำงานทุกๆ วันอาทิตย์ เวลา 1:00 นาฬิกา
175. สั่งให้ทำงานทุกๆ วันที่ 1 ของทุกๆ เดือน เวลา 8:30 นาที
176. จงแสดงชื่อเครื่องที่ใช้งานอยู่ในปัจจุบัน
177. สั่งให้แสดงข้อมูลเฉพาะของการ์ด eth0

178. สั่งให้การ์ด eth0 หยุด/เริ่มการทำงาน
179. กำหนดหมายเลขไอพีแอดเดรสของการ์ดเน็ตเวิร์คให้มีหมายเลขไอพีคือ 192.168.1.100 เน็ตมาร์ค 255.255.255.0
180. กำหนดหมายเลข MAC Address ใหม่ให้กับการ์ด etho ใหม่เป็น 01:02:03:04:05:0A
181. จงตรวจสอบว่า URL www.kernel.org มีหมายเลขไอพีใด
182. จงตรวจสอบว่า URL www.kernel.org มีข้อมูลของ SOA record อะไรบ้าง
183. จงสอบถามายละเอียดการจัดทะเบียนโดเมนของ www.google.com
184. จงตรวจสอบการตอบสนองของโฮสต์ www.google.co.th
185. จงตรวจสอบเส้นทางจากเครื่องของผู้ใช้งานไปยัง www.kernel.org



บทที่ 6

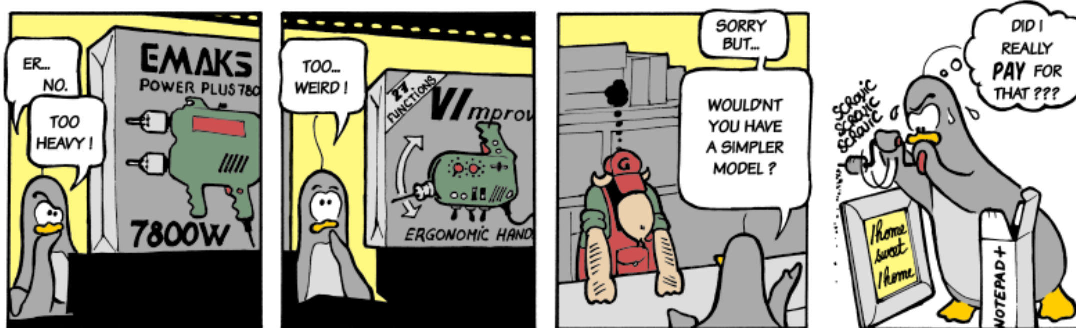
Text Editor (vi/vim)

วัตถุประสงค์

1. สามารถอธิบายการทำงานของ text editor แบบ vi ว่าทำงานอย่างไรได้
2. สามารถใช้ vi สร้าง แก้ไข และจัดการกับแฟ้มข้อมูลได้
3. สามารถใช้ vi แก้ไขหรือเขียนเชลล์สคริปต์ได้

บทที่ 6

Text Editor(vi vs vim)



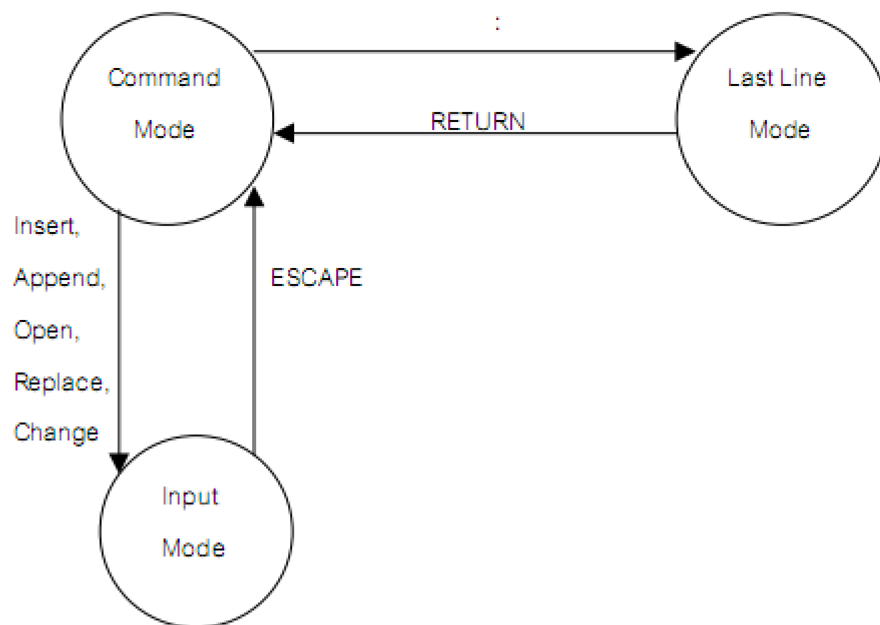
บทนำ

ในระบบยูนิกซ์ทั่วไปจะต้องมีซอฟต์แวร์เอาไว้สำหรับแก้ไขไฟล์เอกสารประจำไว้อยู่ตัวหนึ่งเสมอ ซึ่งก็คือ vi (อ่านว่า วี-ไอ หรือเรียกอีกชื่อว่า vim) vi เป็น text editor [36] ที่มาพร้อมกับระบบปฏิบัติการ Unix ตั้งแต่ยุคต้นๆ ด้วยระบบที่ออกแบบมาใช้งานโดยการคีย์คำสั่งแบบ command line ต้องมีการจำรหัสคำสั่งให้ได้ ไม่มีปุ่มเหมือน text editor สมัยปัจจุบัน แต่มากด้วยประสิทธิภาพ และทำงานได้เร็ว ผู้ใช้งานที่เริ่มใช้งาน vi ในเบื้องต้นจะมีความรู้สึกว่าใช้ยาก และต้องจดจำคำสั่งมาก เป็นภาระในการทำงานเพิ่มขึ้น แต่เมื่อใช้ไปได้สักระยะหนึ่งจะมีความรู้สึกว่าเป็น text editor ที่ดีมากตัวหนึ่ง

โปรแกรม Text Editor vi

โปรแกรม vi แบ่งการทำงานออกเป็น 3 โหมด คือ

1. โหมดคำสั่ง (Command Mode)
2. โหมดพิมพ์ข้อความ (Insert Mode)
3. โหมด lastline



แต่ละโหมดของโปรแกรมจะมีรูปแบบการทำงานและหน้าที่เฉพาะที่แตกต่างกัน ได้แก่

1. **โหมดคำสั่ง** เป็นโหมดที่ใช้ในการควบคุมหรือสั่งการให้โปรแกรม vi ทำงานต่าง ๆ ให้กับผู้ใช้ประกอบด้วย
    - คำสั่งในการเลื่อนเคอเซอร์และหน้าจอ
    - คำสั่งในการเข้าสู่โหมดพิมพ์ข้อความ
    - คำสั่งในการลบตัวอักษร
    - คำสั่งในการลบบรรทัด
    - คำสั่งในการแทนที่ตัวอักษร
    - คำสั่งในการเพิ่มบรรทัดใหม่
    - คำสั่งในการยกเลิกการกระทำล่าสุด เป็นต้น
2. **โหมดพิมพ์ข้อความ** เป็นโหมดที่ใช้ในการพิมพ์หรือป้อนข้อความบนพื้นที่ทำงานของโปรแกรม
    - คำสั่งในการเลื่อนเคอเซอร์และหน้าจอ
    - คำสั่งในการเข้าสู่โหมดพิมพ์ข้อความ
    - คำสั่งในการลบตัวอักษร
    - คำสั่งในการลบบรรทัด
    - คำสั่งในการแทนที่ตัวอักษร
    - คำสั่งในการเพิ่มบรรทัดใหม่
3. **โหมด lastline** เป็นโหมดที่ใช้ในการสั่งงานให้โปรแกรมทำงานดังนี้
    - คำสั่งที่เกี่ยวกับการบันทึกข้อมูลลงไฟล์

- คำสั่งในการค้นหาข้อความ

การเริ่มต้นใช้งานโปรแกรม vi

การเข้าสู่โปรแกรม vi นั้นสามารถทำได้หลายรูปแบบ ดังนี้

รูปแบบที่ 1 พิมพ์คำสั่ง vi file

vi myfile.txt

เป็นคำสั่งให้ทำการเปิดแฟ้มข้อมูลชื่อว่า myfile.txt โดยเริ่มต้นแสดงผลตั้งแต่บรรทัดที่ 1 เมื่อไฟล์ที่ต้องการเปิดไม่มีในระบบ จะทำการสร้างไฟล์ว่างให้ก่อนในเบื้องต้น จะสร้าง myfile.txt ให้ก็ต่อเมื่อมีการบันทึกข้อมูลก่อนออกจากโปรแกรม vi

รูปแบบที่ 2 พิมพ์คำสั่ง vi + file

vi +15 myfile.txt

เป็นคำสั่งให้ทำการเปิดแฟ้มข้อมูลชื่อว่า myfile.txt โดยเริ่มต้นแสดงผลตั้งแต่บรรทัดที่ 15

รูปแบบที่ 3 พิมพ์คำสั่ง vi + file

vi + myfile.txt

เป็นคำสั่งให้ทำการเปิดแฟ้มข้อมูลชื่อว่า myfile.txt โดยเริ่มต้นแสดงผลบรรทัดสุดท้ายของไฟล์

รูปแบบที่ 4 พิมพ์คำสั่ง vi +/pattern file

vi +/Hacker myfile.txt

เป็นคำสั่งให้ทำการเปิดแฟ้มข้อมูลชื่อว่า myfile.txt โดยเริ่มต้นแสดงผลบรรทัดที่ระบุไว้ใน /pattern จากตัวอย่างจะเริ่มเปิดไฟล์ ของบรรทัดที่มีคำว่า Hacker อยู่ด้วย

รูปแบบที่ 5 พิมพ์คำสั่ง vi –r file

vi -r myfile.txt

เป็นคำสั่งให้ทำการกู้คืนแฟ้มข้อมูลชื่อว่า myfile.txt ในกรณีที่กำลังทำงานอยู่แล้วระบบเกิดล้มเหลว (crash)

โหมด lastline

การบันทึกและออกจากไฟล์

เป็นโหมดที่อนุญาตให้ผู้ใช้งานสามารถ ใช้คำสั่งเพิ่มเติมของ vi ได้ โดยการกดปุ่ม " : " (โค-ลอน) ซึ่ง vi จะแสดงเป็นพร้อมต์ รอรับคำสั่งอยู่ด้านล่างสุด ของจอภาพ ผู้ใช้สามารถทำการสั่งโดยการพิมพ์คำว่า " wq " แล้วกด Enter เพื่อทำการบันทึกข้อความลงไฟล์ แล้วออกจากโปรแกรม vi เป็นต้น สำหรับคำสั่งในโหมด lastline แสดงในตารางที่ 6-1

หมายเหตุ: เพื่อให้แน่ใจว่าผู้ใช้ไม่ได้อยู่ในโหมดเพิ่มข้อความ (insert mode) ให้กดปุ่ม ESC สัก 1 ครั้งก่อนเสมอ

ตารางที่ 6-1 คำสั่งการบันทึกและออกจากไฟล์

คีย์(Key)	คำอธิบาย
ESC :e file (กดปุ่ม esc ก่อน จากนั้นจึงกด : e file ตามลำดับ)	ขณะที่ทำงานอยู่ในไฟล์ชื่อ file1 และมีความประสงค์จะนำไฟล์ชื่อ file2 มาแก้ไขเพิ่มเติม เมื่อออกคำสั่ง :e file2 ผลลัพธ์ที่ได้คือจะทำการบันทึกข้อมูลใน file1 ที่กำลังทำงานอยู่ให้ก่อน จากนั้นจึงเปิด file2 ขึ้นมาทำงานต่อไป
ESC :w	ทำการบันทึกข้อมูลในไฟล์ที่กำลังทำงานอยู่ลงบนดิสก์
ESC :w file	ทำการบันทึกข้อมูลแบบเปลี่ยนชื่อใหม่ (save as) ในไฟล์ที่กำลังทำงานอยู่ลงบนดิสก์
ESC :w! file	ทำการบันทึกข้อมูลในไฟล์ที่กำลังทำงานอยู่ลงบนดิสก์เหมือนกับ :w file แต่แตกต่างกันตรงที่ในบางครั้งอาจจะมีการเปิดไฟล์ชื่อเดียวกันอยู่มากกว่า 1 ครั้งทำให้การบันทึกแบบ :w อาจจะไม่สามารถทำได้ แต่ถ้าใช้ :w! file จะพยายามที่จะบันทึกข้อมูลให้ได้
ESC :q	ออกจาก vi
ESC :wq	ออกจาก vi พร้อมกับบันทึกข้อมูล
ESC :wq!	เหมือนกับ :wq แต่ในบางครั้งถ้าไม่สามารถบันทึกและออกได้ ให้ใช้ ! จะบังคับให้โปรแกรมทำการบันทึกและออกจากโปรแกรม
ESC :x	ออกจาก vi พร้อมกับบันทึกข้อมูลเมื่อมีการแก้ไข
ESC :q!	ออกจาก vi และไม่มีการบันทึกข้อมูล

โหมดคำสั่ง

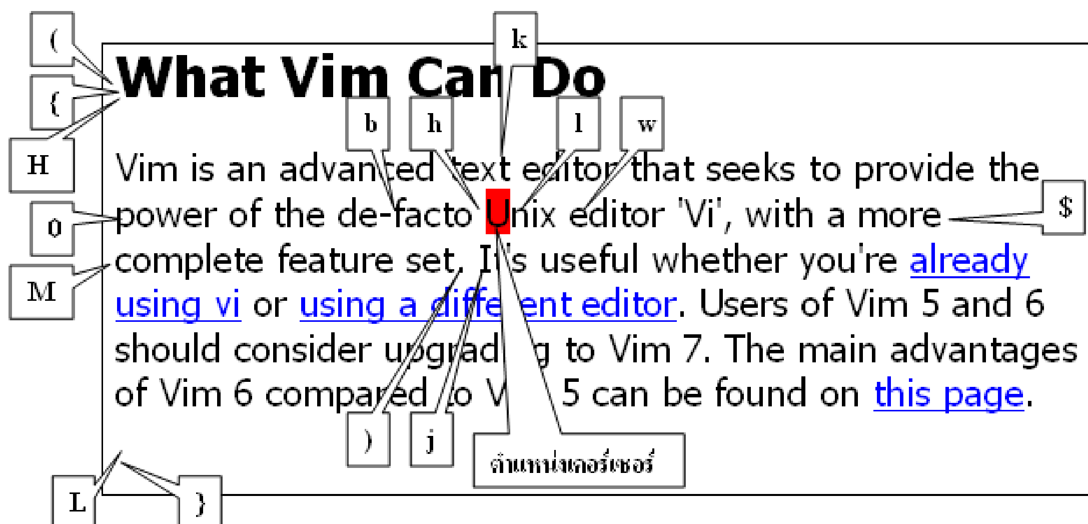
การเลื่อนตำแหน่งเคอเซอร์

ในขณะที่อยู่ในโหมดคำสั่ง ผู้ใช้สามารถเลื่อนเคอเซอร์ไปยังตำแหน่งของอักษรใด ๆ บนหน้าจอได้ และสามารถเลื่อนหน้าจอเพื่อแสดงข้อมูลที่ส่วนใดส่วนหนึ่งของหน้าจอโปรแกรมได้ โดยใช้คำสั่งในการเลื่อนเคอเซอร์ ดังตารางที่ 6-2 สำหรับตัวอย่างการใช้ vi สามารถดูได้จากรูปที่ 6-2

ตารางที่ 6-2 คำสั่งในการเลื่อนเคอเซอร์

คีย์(Keys Pressed)	คำอธิบาย + ผลการทำงาน
l หรือกดปุ่ม **Space**	เลื่อนเคอเซอร์ไปทางขวา 1 ตัวอักษร
H	เลื่อนเคอเซอร์ไปทางซ้าย 1 ตัวอักษร

W	เลื่อนเคอเซอร์ไปทางขวา 1 คำ
B	เลื่อนเคอเซอร์ไปทางซ้าย 1 คำ
K	เลื่อนเคอเซอร์ขึ้นบน 1 บรรทัด
J หรือกดปุ่ม **Enter**	เลื่อนเคอเซอร์ลงล่าง 1 บรรทัด
0 (ศูนย์)	เลื่อนเคอเซอร์ไปต้นบรรทัด
$	เลื่อนเคอเซอร์ไปท้ายบรรทัด
)	เลื่อนเคอเซอร์ไปทางขวา 1 ประโยค(พิจารณา . คือ 1 ประโยค)
(	เลื่อนเคอเซอร์ไปทางซ้าย 1 ประโยค
}	เลื่อนเคอเซอร์ไปย่อหน้าถัดไป
{	เลื่อนเคอเซอร์ไปย่อหน้าก่อน
H	เลื่อนเคอเซอร์ไปยังมุมบนซ้ายของจอ
M	เลื่อนเคอเซอร์ไปยังบรรทัดที่อยู่กึ่งกลางจอ
L	เลื่อนเคอเซอร์ไปยังบรรทัดล่างสุดของจอ
gg	เลื่อนเคอเซอร์ไปบรรทัดแรกสุด
G	เลื่อนเคอเซอร์ไปบรรทัดท้ายสุด
1G	เลื่อนเคอเซอร์ไปบรรทัดแรกสุด
nG	เลื่อนเคอเซอร์ไปบรรทัดที่ระบุใน n เช่น 15G จะไปยังบรรทัดที่ 15
<Control>u	เลื่อนหน้าจอขึ้นครึ่งหน้าจอ (กดปุ่ม ctrl พร้อมกับตัวอักษร u)
<Control>d	เลื่อนหน้าจอลงครึ่งหน้าจอ (กดปุ่ม ctrl พร้อมกับตัวอักษร d)
<Control>f	เลื่อนหน้าจอขึ้นหนึ่งหน้าจอ (กดปุ่ม ctrl พร้อมกับตัวอักษร f)
<Control>b	เลื่อนหน้าจอลงหนึ่งหน้าจอ (กดปุ่ม ctrl พร้อมกับตัวอักษร b)
<Control>L	Refresh หน้าจอ (กดปุ่ม ctrl พร้อมกับตัวอักษร L)
[[	เลื่อนเคอเซอร์ไปยังต้นไฟล์
]]	เลื่อนเคอเซอร์ไปยังท้ายไฟล์



รูปที่ 6-2 ตัวอย่างการใช้ vi

โหมดคำสั่ง

การแก้ไขและแทนที่ข้อความ

เมื่อผู้ใช้งานเลื่อนเคอเซอร์ มายังตำแหน่งที่ต้องการ ถ้าต้องการแก้ไขหรือแทนที่ข้อความลงไปในแฟ้มข้อมูลสามารถใช้คำสั่งได้ ดังตารางที่ 6-3

ตารางที่ 6-3 คำสั่งการแก้ไขหรือแทนที่ข้อความ

คีย์(Keys Pressed)	คำอธิบาย + ผลการทำงาน
cw	พิมพ์ทับข้อความเริ่มตั้งแต่ตำแหน่งที่เคอเซอร์อยู่ครั้งละ 1 คำ
ncw	พิมพ์ทับข้อความเริ่มตั้งแต่ตำแหน่งที่เคอเซอร์อยู่ครั้งละ n คำ เช่น 3cw จะพิมพ์ทับข้อความ 3 คำต่อจากตำแหน่งเคอเซอร์
c$	พิมพ์ทับข้อความเริ่มตั้งแต่ตำแหน่งที่เคอเซอร์อยู่ไปจนถึงคำสุดท้ายของบรรทัด
cc	พิมพ์ทับข้อความในบรรทัดปัจจุบัน
ncc	พิมพ์ทับข้อความ n บรรทัด เริ่มตั้งแต่บรรทัดปัจจุบัน เช่น 3cc จะพิมพ์ทับทั้งหมด 3 บรรทัด
r	พิมพ์ทับตรงตำแหน่งเคอเซอร์ทีละ 1 ตัวอักษร
R	พิมพ์ทับตั้งแต่ตำแหน่งเคอเซอร์ไปเรื่อยๆ จนกว่าจะยกเลิก (ยกเลิกโดยการกด ESC)
S	พิมพ์ทับข้อความในบรรทัดปัจจุบัน
~	สลับตัวอักษรระหว่างตัวอักษรเล็ก(lowercase) และใหญ่(upercase)

โหมดคำสั่ง

การคัดลอกและการวางข้อความ

เมื่อผู้ใช้งานสามารถคัดลอกและวางข้อความได้ด้วยคำสั่ง ดังตารางที่ 6-4

ตารางที่ 6-4 คำสั่งการแก้ไขหรือแทนที่ข้อความ

คีย์(Keys Pressed)	คำอธิบาย + ผลการทำงาน
yy	คัดลอกข้อความทั้งบรรทัด
nyy	คัดลอกข้อความตั้งแต่ตำแหน่งเคอเซอร์ เป็นจำนวน n บรรทัด เช่น 15yy คัดลอกข้อมูลตั้งแต่เคอเซอร์ไปอีก 15 บรรทัด
yw	คัดลอกข้อความ 1 ข้อความ เริ่มตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงอักษรว่าง ถือว่าเป็น 1 ข้อความ
nyw	คัดลอกข้อความตั้งแต่ตำแหน่งเคอเซอร์ เป็นจำนวน n คำ เช่น 15ny คัดลอกข้อมูลตั้งแต่เคอเซอร์ ไปอีก 15 คำ
y$	คัดลอกข้อความ เริ่มตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงท้ายบรรทัด
yG	คัดลอกข้อความ เริ่มตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงท้ายไฟล์
P (เล็ก)	วางข้อมูลที่ได้คัดลอกไว้หลังตำแหน่งเคอเซอร์
P (ใหญ่)	วางข้อมูลที่ได้คัดลอกไว้หน้าตำแหน่งเคอเซอร์
u	ยกเลิกคำสั่งครั้งหลังสุด

โหมดคำสั่ง

การลบข้อความ

เมื่อผู้ใช้งานเลื่อนเคอเซอร์ มายังตำแหน่งที่ต้องการ ถ้าต้องการแก้ไขหรือแทนที่ข้อความลงไปในแฟ้มข้อมูลสามารถใช้คำสั่งได้ ดังตารางที่ 6-5

ตารางที่ 6-5 คำสั่งการแก้ไขหรือแทนที่ข้อความ

คีย์(Keys Pressed)	คำอธิบาย + ผลการทำงาน
x (เล็ก)	ลบอักษรตรงตำแหน่งที่เคอเซอร์อยู่ครั้งละ 1 ตัวอักษร
Nx	ลบอักษรตรงตำแหน่งที่เคอเซอร์อยู่ครั้งละ n ตัวอักษร เช่น 5x ลบตัว

	ตัวอักษรตั้งแต่ตำแหน่งเคอเซอร์ไปด้านขวา 5 ตัวอักษร
X (ใหญ่)	ลบอักษรก่อนตำแหน่งที่เคอเซอร์อยู่
nX (ใหญ่)	ลบอักษรก่อนตำแหน่งที่เคอเซอร์อยู่ n ตัวอักษร เช่น 5X ลบตัวอักษรถัดไปด้ายซ้ายจากตำแหน่งเคอเซอร์อีก 5 ตัวอักษร
dw	ลบข้อความ 1 ข้อความ ที่ตำแหน่งเคอเซอร์ ในกรณีที่เคอเซอร์อยู่ตรงกลางข้อความจะทำการลบแค่ครึ่งหนึ่ง และข้อความ 1 ข้อความจะถูกด้วยช่องว่าง (ช่องว่างก็ถือว่าเป็น 1 ข้อความด้วย)
ndw	ลบข้อความ n ข้อความ ที่ตำแหน่งเคอเซอร์ เช่น 5dw ทำการลบข้อความ 5 ข้อความไปทางด้านขวาของเคอเซอร์
d0 (ดีศูนย์)	ลบจากเคอเซอร์จนถึงต้นบรรทัด
d$	ลบจากตำแหน่งเคอเซอร์จนถึงท้ายบรรทัด
dd	ลบข้อความที่ตำแหน่งเคอเซอร์อยู่ทั้งบรรทัด
ndd	ลบข้อความที่ตำแหน่งเคอเซอร์อยู่ไปด้านล่าง n ทั้งบรรทัด เช่น 5nn
d{	ลบข้อความ ณ ตำแหน่งที่เคอเซอร์อยู่ไปถึงตำแหน่งเริ่มต้นของ paragraph
d}	ลบข้อความ ณ ตำแหน่งที่เคอเซอร์อยู่ไปถึงตำแหน่งสิ้นสุดของ paragraph
:1,. d	ลบข้อความ ณ ตำแหน่งที่เคอเซอร์อยู่ไปถึงตำแหน่งเริ่มต้นของไฟล์
:.,$ d	ลบข้อความ ณ ตำแหน่งที่เคอเซอร์อยู่ไปถึงตำแหน่งสุดท้ายของไฟล์
:1,$ d	ลบข้อความทั้งหมดในไฟล์

โหมดคำสั่ง

การมาร์คตำแหน่งและกำหนดขนาดบัพเฟอร์

ผู้ใช้งานสามารถกำหนดจุดทำงาน(markers) ในไฟล์เพื่อความสะดวกในกรณีที่ต้องเคลื่อนย้ายหรือเปลี่ยนตำแหน่งไปทำงานในส่วนใดส่วนหนึ่งของโปรแกรมบ่อยๆ หรือกำหนดขนาดพื้นที่ในการใช้เก็บข้อมูลชั่วคราว(buffers) ในขณะทำงานได้ ดังตารางที่ 6-6

ตารางที่ 6-6 คำสั่งการมาร์คตำแหน่งและขนาดบัพเฟอร์

คำสั่ง(Command)	คำอธิบาย + ผลการทำงาน
mf	กำหนดตำแหน่งที่มาร์คไว้ ในตำแหน่งที่เคอเซอร์อยู่ ในการกำหนดดังกล่าวจะไม่มีสัญลักษณ์แสดงให้เห็นว่ามีการมาร์คไว้ ทดสอบได้โดยการเลื่อนเคอเซอร์ไปยังตำแหน่งอื่น จากนั้นให้กดปุ่ม `f เคอเซอร์จะเลื่อน

	ไปยังตำแหน่งที่มาร์คไว้ทันที
`f	เลื่อนเคอเซอร์ไปยังตำแหน่งที่มาร์คไว้
´f	เลื่อนเคอเซอร์ไปยังบรรทัดที่มาร์คไว้
"s12yy	สร้างบัฟเฟอร์ใหม่ชื่อว่า s และทำการคัดลอกข้อมูลจำนวน 12 บรรทัดไปยังบัฟเฟอร์ดังกล่าว
"ty}	คัดลอกข้อมูลตั้งแต่เคอเซอร์อยู่ถึงจุดสิ้นสุดของ paragraph และไปเก็บไว้ในบัฟเฟอร์ชื่อว่า t
"ly1G	คัดลอกข้อมูลตั้งแต่เคอเซอร์อยู่ถึงจุดเริ่มต้นของไฟล์ และไปเก็บไว้ในบัฟเฟอร์ชื่อว่า l
"kd`f	คัดลอกข้อมูลตั้งแต่เคอเซอร์อยู่ถึงจุดที่มาร์คไว้ และไปเก็บไว้ในบัฟเฟอร์ชื่อว่า k
"kp	เขียนข้อมูลในบัฟเฟอร์ไปยังไฟล์

โหมดคำสั่ง

การค้นหาและค้นหาแบบแทนที่ข้อความ

ผู้ใช้งานสามารถค้นหาข้อความในแฟ้มข้อมูล ด้วยคำสั่ง ดังตารางที่ 6-7

ตารางที่ 6-7 การค้นหาข้อความ

คำสั่ง(Command)	คำอธิบาย + ผลการทำงาน
/String	ค้นหาข้อมูลที่ต้องการ ทิศทางในการค้นหาจะเริ่มตั้งแต่ตำแหน่งเคอเซอร์อยู่ ไปยังท้ายของไฟล์(ค้นหาต่อไปเรื่อยๆ ให้กดปุ่ม n) ตัวอย่าง /software โปรแกรมจะทำการค้นหาคำว่า software จากบนลงล่าง
?String	ค้นหาข้อมูลที่ต้องการ ทิศทางในการค้นหาจะเริ่มตั้งแต่ตำแหน่งเคอเซอร์อยู่ ไปยังจุดเริ่มต้นของไฟล์ ตัวอย่าง /software โปรแกรมจะทำการค้นหาคำว่า software จากล่างขึ้นบน
/^String	เลื่อนเคอเซอร์ไปยังบรรทัดพบข้อความที่ต้องการค้นหา เช่น /^The จะค้นหาทุกคำที่มีตัวอักษร The ประกอบด้วย เช่น The, Then, There
/^String\>	เลื่อนเคอเซอร์ไปยังบรรทัดพบข้อความที่ต้องการค้นหา แต่ข้อความที่ค้นหาจะต้องเป็นคำที่ต้องการเท่านั้น เช่น /^The\> จะค้นหาคำว่า The เท่านั้น
/[bB]ox	[bB] จะกำหนดเงื่อนไขในการค้นหาว่าเป็นอักษรตัวบี แบบเล็กหรือใหญ่

	ก็ได้ และตามด้วย ox เท่านั้นคำที่ค้นหาได้จะพบคำว่า box หรือ Box ก็ได้
n	ค้นหาต่อไปเรื่อยๆ จนกว่าจะพบคำที่ต้องการ
N	ค้นหาต่อไปเรื่อยๆ แบบกลับทางกับ n จนกว่าจะพบคำที่ต้องการ
:s/String1/String2/g	ค้นหาคำที่กำหนดใน String1 เมื่อพบจะทำการแทนที่คำด้วย String2 แทน เช่น :s/UNIX/Linux/g จากตัวอย่างจะทำการแทนที่คำว่า UNIX ด้วย Linux

โหมดคำสั่ง

การกำหนดรูปแบบการค้นหา

ผู้ใช้งานสามารถกำหนดรูปแบบในการค้นหาข้อความได้ โดยอาศัยสัญลักษณ์ต่างๆ ที่ vi กำหนดให้ ดังตารางที่ 6-8

ตารางที่ 6-8 สัญลักษณ์ที่ใช้ในการสร้างรูปแบบการค้นหา

Expression	คำอธิบาย + ผลการทำงาน
.	ตัวอักษรอะไรก็ได้เพียง 1 ตัวอักษร เช่น T.e ข้อความที่เป็นไปได้คือ Tae, The, Tee เป็นต้น
*	ตัวอักษรอะไรก็ได้ กี่ตัวก็ได้ รวมถึงตัวอักษรว่างด้วย
.*	เหมือนกับ *
\<	กำหนดตำแหน่งเริ่มต้นของข้อความที่ต้องการค้นหา เช่น /\<The
\>	กำหนดตำแหน่งสิ้นสุดของข้อความที่ต้องการค้นหา เช่น /\<The\>
\	คำสั่งที่อนุญาติให้เพิ่มอักษรพิเศษเข้าไปได้ เช่น \\ ต้องการค้นหาข้อความที่มีอักษร \ อยู่ในข้อความด้วย, * ต้องการค้นหาข้อมูลที่มี *
*	ค้นหาต่อไปเรื่อยๆ แบบกลับทางกับ n จนกว่าจะพบคำที่ต้องการ
^	จุดเริ่มต้นของแต่ละบรรทัด
$	จุดสิ้นสุดของแต่ละบรรทัด
[set]	ค้นหาตัวอักษรตัวใดตัวหนึ่งที่อยู่ในเซ็ต เช่น [bB]ox ข้อความที่ค้นหาได้จะมีรูปแบบเป็น box หรือ Box ก็ถูกต้องทั้งคู่
[^*set*]	ค้นหาตัวอักษรตัวใดตัวหนึ่งที่ไม่อยู่ในเซ็ต เช่น [bB]ox ข้อความที่ค้นหาได้จะต้องไม่มีรูปแบบเป็น box หรือ Box ข้อความที่ถูกคือ aox, cox เป็นต้น

[^XYZ[:digit:]]	ข้อความอะไรก็ได้ยกเว้น X, Y และ Z หรือตัวเลข ข้อความที่ถูกต้องคือ xyz, abc, aa เป็นต้น

โหมดพิมพ์ข้อความ

การพิมพ์ข้อความ

เมื่อผู้ใช้งานเลื่อนเคอเซอร์ มายังตำแหน่งที่ต้องการ ในหัวข้อการเลื่อนตำแหน่งเคอเซอร์แล้ว ถ้าต้องการพิมพ์หรือเพิ่มข้อความลงไปในแฟ้มข้อมูลสามารถใช้คำสั่งได้ ดังตารางที่ 6-9

ตารางที่ 6-9 คำสั่งการพิมพ์หรือเพิ่มข้อความ

คีย์(Keys Pressed)	คำอธิบาย + ผลการทำงาน
a	พิมพ์/เพิ่ม ข้อมูลต่อจากตำแหน่งที่เคอเซอร์อยู่
A	พิมพ์/เพิ่ม ข้อมูลต่อจากท้ายบรรทัดที่เคอเซอร์อยู่
i	พิมพ์/เพิ่ม ข้อมูลหน้าตำแหน่งที่เคอเซอร์อยู่
I	พิมพ์/เพิ่ม ข้อมูลต้นบรรทัดที่เคอเซอร์อยู่
o	แทรกบรรทัดด้านล่างที่เคอเซอร์อยู่
O	แทรกบรรทัดด้านบนที่เคอเซอร์อยู่

บทสรุป

Text Editor ที่นิยมใช้งานบนระบบปฏิบัติการยูนิกซ์และลินุกซ์นั้นมีให้เลือกหลายตัว เช่น vi, emacs, pico, gedit, nano เป็นต้น แต่ vi เป็น text editor ที่เก่าแก่และติดตั้งมาพร้อมกับระบบปฏิบัติการยูนิกซ์และลินุกซ์อยู่แล้ว รวมทั้งมีประสิทธิภาพการใช้งานที่ดีมากตัวหนึ่ง และเหมาะสมที่จะใช้ในการเขียนโปรแกรมสคริปต์ต่อไป ดังนั้นจึงมีความจำเป็นต้องเรียนรู้และใช้งาน vi ให้เข้าใจ เพื่อใช้กับงานประยุกต์ระบบปฏิบัติการในอนาคตได้อย่างมีประสิทธิภาพ

คำถามท้ายบท

1. แสดงคำสั่งการเรียกใช้ vi เพื่อ edit file ชื่อ data.dat

   ข้อความใน data.dat ดังนี้

What Vim Can Do

Vim is an advanced text editor that seeks to provide the power of the de-facto Unix editor 'Vi', with a more complete feature set. It's useful whether you're already using vi or using a different editor. Users of Vim 5 and 6 should consider upgrading to Vim 7. The main advantages of Vim 6 compared to Vim 5 can be found on this page.

A General Overview

Copyright (c) 2007 Laurent Gregoire

What Is Vim?

Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems.

Vim is often called a "programmer's editor," and so useful for programming that many consider it an entire IDE. It's not just for programmers, though. Vim is perfect for all kinds of text editing, from composing email to editing configuration files.

Despite what the above comic suggests, Vim can be configured to work in a very simple (Notepad-like) way, called evim or Easy Vim.

What Vim Is Not?

Vim isn't an editor designed to hold its users' hands. It is a tool, the use of which must be learned.

Vim isn't a word processor. Although it can display text with various forms of highlighting and formatting, it isn't there to provide WYSIWYG editing of typeset documents. (It is great for editing TeX, though.)

Vim's License

Vim is charityware. Its license is GPL-compatible, so it's distributed freely, but we ask that if you find it useful you make a donation to help children in Uganda through the ICCF. The full license text can be found in the documentation. More information about charityware on Charity-ware.org.

Vim in Six Kilobytes

If all of this information is overwhelming, try a smaller dose. We can expound

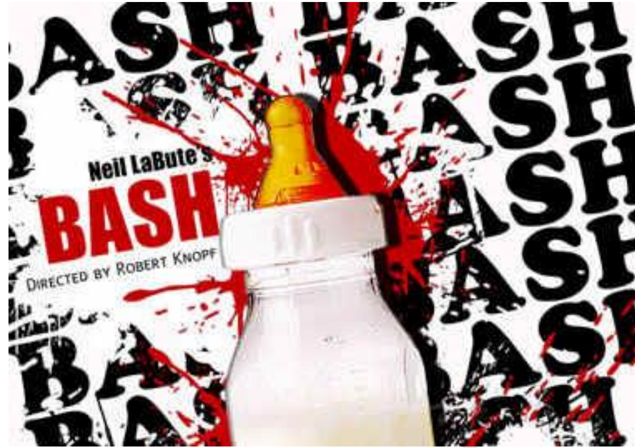
the wonders of vim in just six kilobytes -- and in more languages than you can shake a stick at!

Testimonials

Don't take our word for it! Read what others have said about Vim.

2. ถ้าอยู่ใน command mode แสดงคำสั่งเพื่อเข้าสู่ insert mode ณ ตำแหน่งของ cursor
3. ถ้าอยู่ใน command mode แสดงคำสั่งเพื่อเข้าสู่ insert mode ณ ท้ายบรรทัดปัจจุบัน
4. ถ้าอยู่ใน insert mode แสดงคำสั่งที่จะเข้าสู่ command mode
5. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ delete character ณ ตำแหน่งของ cursor
6. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะลบบรรทัดที่ cursor อยู่
7. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะลบบรรทัดที่ cursor อยู่และบรรทัดถัดไปอีก 5 บรรทัด
8. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ save การเปลี่ยนแปลงของ file และออกจาก editor
9. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะยกเลิกการเปลี่ยนแปลงของ file และออกจาก editor
10. ถ้าอยู่ใน insert mode แสดงคำสั่งที่จะ save การเปลี่ยนแปลงของ file และออกจาก editor
11. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะเลื่อน cursor ขึ้น 1 บรรทัด
12. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะเลื่อน cursor ลง 1 บรรทัด
13. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะเลื่อน cursor ไปทางขวา 1 ตัวอักษร
14. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะเลื่อน cursor ไปทางซ้าย 1 ตัวอักษร
15. ถ้าอยู่ใน command mode แสดงคำสั่งที่ไปยังบรรทัดแรกของ file
16. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะค้นหา string “Vim“
17. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ copy บรรทัดที่ cursor อยู่และอีก 5 บรรทัดถัดไปลงสู่ clipboard
18. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ copy บรรทัดที่อยู่ใน clipboard มาไว้ก่อนบรรทัดที่ cursor อยู่
19. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ copy บรรทัดที่อยู่ใน clipboard มาไว้ถัดจากบรรทัดที่ cursor อยู่
20. ถ้าอยู่ใน input mode แสดงวิธีที่จะขึ้นบรรทัดใหม่
21. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะลบ character ใต้ cursor พร้อมกับอีก 5 characters ถัดไป

22. ถ้าอยู่ใน command mode หากเผลอทำการ delete บรรทัดไป 12 บรรทัดในคราวเดียวกัน แสดงวิธีที่จะ undelete บรรทัดเหล่านั้นกลับคืน
23. ถ้าอยู่ใน command mode แสดงวิธีที่จะเชื่อมบรรทัดที่ cursor อยู่กับบรรทัดถัดไป
24. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ move forward ไปทั้ง screen
25. ถ้าอยู่ใน command mode แสดงคำสั่งที่จะ move backward กลับมาทั้ง screen



บทที่ 7

การเขียนโปรแกรมเชลล์สคริปต์

วัตถุประสงค์

1. สามารถการเขียนโปรแกรมเชลล์สคริปต์บนระบบปฏิบัติการยูนิกซ์/ลินูกซ์ได้
2. สามารถเขียนโปรแกรมเชลล์สคริปต์ได้อย่างถูกต้องและมีประสิทธิภาพ
3. สามารถเขียนโปรแกรมเชลล์สคริปต์แก้ไขปัญหาทางระบบคอมพิวเตอร์ได้
4. สามารถประยุกต์ใช้เชลล์สคริปต์ให้สามารถตอบสนองต่องานของผู้ดูแลระบบได้

บทที่ 7
การเขียนโปรแกรมเชลล์สคริปต์
(Shell Script Programming : Bash)

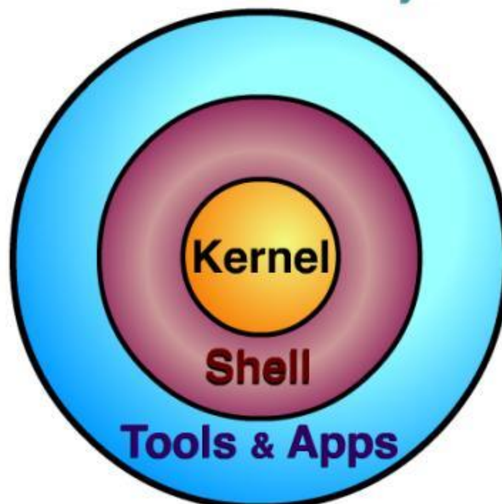
บทนำ

ในบทนี้กล่าวถึงการเขียนโปรแกรม เชลล์สคริปต์ เพื่อใช้สำหรับช่วยควบคุมการทำงานต่างๆ บนระบบปฏิบัติการ เช่น การสำรองข้อมูล การตรวจสอบแฟ้ม การตั้งเวลาการทำงาน การตรวจสอบโครงสร้างของไฟล์ เป็นต้น ซึ่งต้องรวบรวมเอาคำสั่งต่างๆ ที่เรียนรู้จากบทที่ 5 มาประยุกต์เข้าด้วยกัน โดยกำหนดเป้าหมายของการทำงานไว้อย่างชัดเจน ดังนั้นบทนี้จึงเป็นบทที่สำคัญ ซึ่งควรศึกษาหลักการเขียนโปรแกรมเชลล์สคริปต์ เพื่อนำไปประยุกต์ใช้งานในอนาคตต่อไป

การเขียนโปรแกรมเชลล์สคริปต์ [37-45]

โครงสร้างพื้นฐานการทำงานของระบบยูนิกซ์/ลินุกซ์ มีอยู่ 4 ส่วนด้วยกัน คือ ฮาร์ดแวร์ (Hardware), เคอร์เนล (Kernel), เชลล์ (Shell) และแอพพลิเคชัน (Application) ดังรูปที่ 7-1

Parts of the UNIX System



รูปที่ 7-1 แสดงโครงสร้างพื้นฐานการทำงานของระบบยูนิกซ์

เชลล์ (Shell) คือ โปรแกรมหนึ่งบนระบบยูนิกซ์/ลินุกซ์ ที่ทำหน้าที่เป็นส่วนติดต่อผู้ใช้ (interface) ระหว่างผู้ใช้กับระบบปฏิบัติการยูนิกซ์/ลินุกซ์ (เคอร์เนล) ซึ่งเชลล์ไม่ได้เป็นส่วนหนึ่งของเคอร์เนล แต่ใช้เคอร์เนลในการประมวลผล ผู้ใช้สามารถสั่งงานระบบปฏิบัติการได้โดยผ่านทาง

เชลล์เท่านั้น โปรแกรมเชลล์ยังมีคุณสมบัติของ Shell Programming Language ทำให้ผู้ใช้สามารถนำคำสั่งต่างๆของเชลล์มาเขียนเป็นโปรแกรมเก็บเป็นไฟล์ไว้ได้ เรียกว่า เชลล์สคริปต์ (Shell Script)

ประเภทของเชลล์ที่นิยมใช้ในปัจจุบัน

- **Bourne shell** (/bin/sh) เป็นเชลล์ในยุคแรกๆ ที่มีใช้กันอย่างแพร่หลาย มีการกำหนดโครงสร้างภาษาคล้ายๆ กับภาษาอัลกอ (Algo Language) สามารถเขียนเป็น shell script ได้ และยังเป็น เชลล์มาตราฐานที่มีในระบบปฏิบัติการยูนิกซ์ทุกตัว และยังสามารถย้าย shell script ไปยังยูนิกซ์ระบบอื่นได้ โดยไม่ต้องแก้ไขสคริปต์ จะมี default prompt เป็นเครื่องหมาย " $ "
- **C shell** (/bin/csh) เป็นเชลล์ที่พัฒนาขึ้นมาหลังจาก Bourne shell มีรูปแบบคำสั่งและไวยากรณ์เหมือนกับภาษาซี (C Language) มีฟังก์ชันการทำงานหลากหลาย สะดวก อีกทั้งยังสามารถควบคุมการไหลของข้อมูลได้ดีกว่า Bourne shell และยังมีความสามารถในการเรียกใช้คำสั่งที่ใช้ไปแล้ว จะมี default prompt เป็นเครื่องหมาย " % "
- **Korn shell** (/bin/ksh) เป็น shell ที่พัฒนามาจากต้นแบบของ Bourne shell และ C shell สามารถทำงานใน function ของ Bourne shell ได้ทุกอย่าง การเขียน shell script ทำได้ง่ายและรัดกุมขึ้น สามารถนำคำสั่งที่ใช้ไปแล้วกลับมา execute ใหม่ได้ ถือได้ว่า Korn shell เป็นการรวมเอาข้อดีของ Bourne shell และ C shell มาไว้ด้วยกัน แต่ไม่ได้มีใน UNIX ทุกตัว จะมี default prompt เป็นเครื่องหมาย " $ "
- **Bourne again shell** (/bin/bash หรือ /usr/local/bin/bash) เป็นการเอา Bourne shell นำกลับมาพัฒนาใหม่ สามารถทำงานแบบ line editing ได้ และยังได้เพิ่มประสิทธิภาพในการทำงานอีกหลายอย่าง bash shell นี้ไม่ใช่มาตรฐานของยูนิกซ์เชลล์ แต่เป็น default shell ของ linux ในปัจจุบัน จะมี default prompt เป็นเครื่องหมาย " # หรือ $ "

การแสดงผลประเภทของเชลล์ทั้งหมดในระบบปฏิบัติลินุกซ์

ผู้ใช้สามารถตรวจสอบดูประเภทของเชลล์ทั้งหมดที่ระบบปฏิบัติการมี ได้โดยใช้คำสั่ง

```
# cat /etc/shells
/bin/sh
/bin/bash
/bin/tcsh
/bin/csh
/bin/ksh
```

การแสดงประเภทของเชลล์ที่กำลังใช้อยู่ในปัจจุบัน

ผู้ใช้สามารถตรวจสอบดูประเภทของเชลล์ที่กำลังใช้งานอยู่ในขณะนั้น ได้โดยใช้คำสั่ง

```
#echo $SHELL
```

/bin/bash

เป้าหมายของการเขียนเชลล์สคริปต์

สาเหตุที่จำเป็นต้องเขียนเชลล์สคริปต์นั้น สามารถจำแนกได้ดังต่อไปนี้

- สามารถรวบรวมเอาคำสั่งบนระบบยูนิกซ์/ลินุกซ์ มาทำงานร่วมกันเป็นโปรแกรม สำหรับแก้ไขปัญหา หรือตอบสนองการทำงานของผู้ใช้ได้
- คำสั่งต่างๆ บนระบบปฏิบัติการยูนิกซ์/ลินุกซ์ มีความใกล้ชิดกับเคอร์เนลทำให้สามารถประมวลผลได้อย่างรวดเร็ว
- เชลล์สคริปต์สามารถรับข้อมูลเข้า (Input) จากผู้ใช้ ไฟล์ และแสดงผลลัพธ์ออกทางหน้าจอได้เหมือนกับการใช้คำสั่ง (Command) โดยตรง
- มีประโยชน์อย่างมากในการสร้างคำสั่งที่เป็นส่วนบุคคล
- ลดเวลาในการป้อนคำสั่ง เมื่องานดังกล่าวใช้เป็นประจำเสม่ำเสมอ
- สามารถสั่งให้ทำงานบางอย่างได้โดยอัตโนมัติ

ขั้นตอนในการเขียนเชลล์สคริปต์

ขั้นตอนที่ 1: ใช้โปรแกรมประเภท editor ตัวใดตัวหนึ่งเขียน Shell Script เช่น vi, pico, emacs, gedit, nano เป็นต้น การตั้งชื่อสคริปต์สามาถใช้ตัวอักษรเล็กหรือใหญ่ จะให้ความหมายที่แตกต่างกัน เช่น test, Test จะถือว่าเป็นคนละไฟล์กัน หรือสามารถตั้งชื่อโดยใช้ตัวเลขผสมกับสัญลักษณ์คือ -, _ , $ ร่วมด้วย ก็ได้ เช่น 1, 1.txt, 1a, 1_a, Ex-1, file$ เป็นต้น สำหรับส่วนขยายยูนิกซ์จะไม่สนใจว่ามีส่วนขยายเป็นอะไร ถ้ากำหนดสิทธิ์ให้สามารถประมวลผลได้สคริปต์จะสามารถทำงานได้ทั้งหมด เช่น test.txt, test.out, test.dat, test.exe, test.doc เป็นต้น

ขั้นตอนที่ 2: เพิ่มสิทธิให้ไฟล์สคริปต์ที่เขียนขึ้นนั้นสามารถประมวลผล (execute) ได้ด้วยคำสั่ง

```
# chmod +x  myscript
# ls –l
-rwxr-xr-x 1 root root 0 Jan 22 20:03 myscript
```

หรือ

```
$ chmod 755 myscript
```

ขั้นตอนที่ 3: ประมวลผลไฟล์สคริปต์ ดังกล่าว ด้วยคำสั่ง

$ sh myscript ในกรณีที่เป็น Bourne shell

bash myscript ในกรณีที่เป็น Bourne again shell

./myscript

เริ่มต้นเขียนเชลล์สคริปต์ Hello World

สร้างไฟล์เชลล์สคริปต์ด้วยคำสั่ง vi HelloWorld.bash

```
#     ← ①
# My first shell script (Hello World)    ← ②
#
#!/bin/bash  ← ③
clear   ← ④
echo "Hello World !!!"    ← ⑤
```

① สัญลักษณ์ # คือ comment ใช้สำหรับอธิบายการทำงานของเชลล์สคริปต์ เมื่อคอมไพเลอร์ประมวลผลเจอสัญลักษณ์ดังกล่าวจะไม่มีการประมวลผลในบันทัดดังกล่าว

② ข้อความที่ใช้อธิบายความหมายของเชลล์สคริปต์ที่เขียนขึ้น

③ "#!" เรียกว่า sha-bang เป็นสัญลักษณ์บอกว่าไฟล์นี้เป็นสคริปต์และใช้ /bin/bash เป็นตัวแปรภาษา (interpreter)

④ คำสั่งเคลียร์หน้าจอภาพ

⑤ พิมพ์ข้อความว่า Hello World !!! ออกหน้าจอภาพ

```
# chmod +x HelloWorld.bash
# ./HelloWorld.bash
Hello World !!!
```

สังเกตุว่าการสั่งเชลล์สคริปต์ให้ทำงานจะต้องใช้ ./ นำหน้าชื่อเชลล์ ทั้งนี้เนื่องจากไดเรกทอรีปัจจุบันซึ่งได้แก้เครื่องหมาย "." ไม่ได้อยู่ในตัวแปรสภาพแวดล้อม (environment variable) PATH ตัวแปรสภาพแวดล้อม PATH เป็นตัวแปรที่เก็บไดเรกทอรีต่างๆ ขั้นด้วยเครื่องหมาย : บอกให้เชลล์รู้ว่าโปรแกรมที่ใช้ได้อยู่ในระบบอยู่ในไดเรกทอรีเหล่านี้ กล่าวคือเชลล์จะใช้ตัวแปรสภาพแวดล้อม PATH ในการหาว่าคำสั่ง (โปรแกรม) อยู่ที่ไหน ดังนั้นถ้ารันเชลล์สคริปต์โดยไม่ใส่เครื่องหมาย "./" จะหมายความว่า "HelloWorld.bash" เป็นโปรแกรมคำสั่งที่อยู่ในไดเรกทอรีที่กำหนดไว้ในตัวแปรสภาพแวดล้อม PATH ซึ่งจริงๆแล้ว "." (ไดเรกทอรีปัจจุบัน) ไม่ได้อยู่ใน PATH จึงต้องอ้างอิงสคริปต์โดยใช้ relative path คือ "./" บอกกับเชลล์ว่า HelloWorld.bash อยู่ที่นี่ไดเรกทอรีที่ทำงานอยู่ (.)

ตัวแปรสภาพแวดล้อม (environment variable) PATH

```
# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

อีกวิธีหนึ่งสำหรับการรันเชลล์สคริปต์ คือ การเรียกใช้โปรแกรม bash แปลคำสั่งที่อยู่ในไฟล์ HelloWorld.bash ส่วนแบบที่สองคือการใช้คำสั่ง "." อ่านเนื้อหาในไฟล์ HelloWorld.bash แล้วจึงตีความหมาย

```
# bash HelloWorld.bash
# . HelloWorld.bash
```

ตัวแปร (Variables)

ตัวแปรในเชลล์สคริปต์แบ่งออกเป็น 2 ประเภทคือ ตัวแปรระบบ (System variables ใช้อักษรตัวใหญ่ เช่น $PATH, $SHELL) และตัวแปรที่ผู้ใช้สร้างขึ้นเอง (User defined variables ควรใช้อักษรตัวเล็ก) เพื่อใช้สำหรับเก็บค่าต่างๆ ตัวแปรที่ใช้ในเชลล์ไม่จำเป็นต้องประกาศประเภทของตัวแปรเหมือนภาษาซี (เช่น int, float) และสามารถตั้งค่าใช้ได้ทันที การตั้งชื่อตัวแปรต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมายขีดล่าง (Underscore , _) ที่เหลือจะเป็นตัวอะไรก็ได้ เช่น name="Mr. Somsak Srisai" คือการกำหนดให้ตัวแปร name เก็บค่าสตริง Mr. Somsak Srisai

ตัวแปรที่ผู้ใช้สร้างขึ้นเอง (User defined variables)

การตั้งค่าตัวแปรจะมีรูปแบบดังนี้

ชื่อตัวแปร=ค่าตัวแปร

```
# var1=Hello
# var2=world
```



ข้อควรระวัง คือช่วงก่อนหน้าและหลังเครื่องหมายเท่ากับ ห้ามมีช่องว่าง เพราะเชลล์ถือว่าช่องว่าคือตัวแบ่งอาร์กิวเมนต์

การแสดงค่าที่กำหนดให้ใช้เครื่องหมายดอลล่าร์ ("$") นำหน้าตัวแปรที่ต้องการ

```
# echo $var1 $var2
Hello World
```

มีบางกรณีที่เราต้องการอ้างอิงตัวแปร แล้วเขียนข้อความต่อจากตัวแปรเลย เช่น

```
# temp=Myfile
# echo $temp.txt
Myfile.txt
```

ในกรณีนี้จะไม่มีปัญหาในการตีความ เพราะเชลล์ตีความตัวแปรถึงแค่ เครื่องหมาย "." เท่านั้น เชลล์จะแทนค่า "$temp" ด้วย "Myfile" จากนั้นจะพิมพ์คำว่า ".txt" ตามหลังทันที แต่คำที่ต่อจากตัวแปรไม่ใช่จุดเช่น "_Script" เชลล์จะตีความว่าเป็นการอ้างอิงถึงตัวแปรชื่อ "Myfile_Script" ซึ่งตัวแปรดังกล่าวไม่มีอยู่ในระบบ เพื่อขจัดความคลุมเครือในกรณีดังกล่าวให้ใช้เครื่องหมายวงเล็บปีกกา "{}" ช่วยระบุช่วงของตัวแปรที่ต้องการแสดง ตัวอย่างเช่น

```
# temp=Myfile
# echo ${temp}_Script
Myfile_Script
```

Quote และ Double quote

ช่องว่าง (space) มีความหมายพิเศษสำหรับเชลล์ไว้สำหรับแบ่งอาร์กิวเมนต์ ดังนั้นถ้าต้องการตั้งค่าตัวแปรที่มีช่องว่าอยู่ด้วยเช่น "Hello world" ต้องใช้เครื่องหมายคำพูด quote (') หรือ double quote (") คล่อมคำที่ต้องการ

```
# var3='Hello world'
# var4="Hello world"
# echo $var3 $var4
Hello World Hello World
```

การใช้เครื่องหมาย ‘ และ “ แตกต่างกันในกรณีที่สั่งพิมพ์ค่าของตัวแปรแทน คือ

```
# echo '$var3 $var4'
# echo "$var3 $var4"
$var3 $var4
Hello World
```



เมื่อต้องการประกาศตัวแปร ที่ไม่มีค่าหรือค่า NULL variable สามารถทำได้โดย ไม่ต้องใส่ค่าเริ่มต้นให้กับตัวแปร **$null_vaule=**

ตัวแปรระบบชนิดบิวท์อิน (System Built in Variables)

ตัวแปรระบบชนิดบิวท์อินเป็นตัวแปรที่ระบบสร้างขึ้นมาแบบอัตโนมัต มีหน้าที่เก็บค่าของตัวแปรต่างๆ ขณะเชลล์กำลังทำงาน ดังตารางที่ 7-1

ตารางที่ 7-1 ตัวแปรที่ระบบสร้างขึ้นเพื่อช่วยในการเขียนโปรแกรม

Shell Built in Variables	คำอธิบาย
$0 ($ศูนย์)	ตัวแปรที่เก็บชื่อโปรแกรม หรือ สคริปต์ที่ทำการเรียกใช้อยู่ปัจจุบัน เรียกว่า position parameter
$1...$9	ตัวแปรที่เก็บค่าอาร์กิวเมนต์ (ค่าที่ถูกส่งผ่านเข้ามาจากโปรแกรมขณะทำการรัน เพื่อนำไปใช้งาน) ของโปรแกรม ซึ่งค่าเหล่านี้ถูกกำหนดตามหลังมากับชื่อโปรแกรม เช่น # hello john (john จะเก็บใน $1) หรือ #hello john max pop amp ($1=john, $2=max, $3=pop, $4=amp)
$*	ตัวแปรที่เก็บค่าอาร์กิวเมนต์ของโปรแกรมทั้งหมด ซึ่งหมายถึง $1 ... $9
$#	เก็บจำนวนของอาร์กิวเมนต์ทั้งหมด
$$	เก็บค่าโพรเซส (PID) ของโปรแกรมที่เรียกใช้อยู่ปัจจุบัน
$!	เก็บค่าตัวเลขโพรเซส (PID) ของโปรแกรมที่รันอยู่เบื้องหลังล่าสุด

$?	เก็บค่าที่ถูกส่งคืนมาจากโปรแกรม หรือคำสั่งที่เพิ่งทำงานเสร็จ (Return Code) ถ้าทำงานจบโดยบริบูรณ์จะมีค่าเป็น 0 ถ้าจบการทำงานโดยมี error ก็จะเป็นตัวเลขที่ไม่ใช่ 0 แล้วแต่กรณี ตัวแปรนี้เอาไว้ตรวจสอบว่าโปรแกรมที่สั่งไปทำงานถูกต้องหรือไม่
$@	คล้ายกับ $* คือ เก็บค่าอาร์กิวเมนต์ของโปรแกรมทั้งหมด แต่จะใช้ช่องว่างคั่นระหว่าง position parameter

ตัวอย่างการใช้ตัวแปร built in

จากตัวอย่างเป็นการสร้างเชลล์สคริปต์ชื่อ calRect เพื่อคำนวณหาพื้นที่สี่เหลี่ยมผืนผ้า โดยรับค่าอินพุต 2 ค่า คือ ด้านกว้าง (width) และยาว (length)

```
# shell script for Rectangule calculation
#!/bin/bash
clear
echo "#########################"
echo "#    Rectangular Shell Script         #"
echo "#########################"
echo "How to use: ./calRect width lenght   #"
width=0 ①
length=0
result=$(($1 * $2)) ②
echo "> result of calRect = $result"
echo "\$0 = $0" ③
echo "\$1 = $1" ④
echo "\$2 = $2" ⑤
echo "\$* = $*" ⑥
echo "\$# = $#" ⑦
echo "\$$ = $$" ⑧
echo "\$! = $!" ⑨
echo "\$? = $?" ⑩
echo "\$@ = $@" ①①
```

สั่งเชลล์สคริปต์ calRect ให้ทำงาน

```
# ./calRect 3 4
```

ผลการทำงานของ

```
#########################
#   Rectangular Shell Script          #
#########################
How to use: ./calRect width length     #
result of calRect = 12
$0 = ./calRect
$1 = 3
$2 = 4
$* = 3 4
```

```
$# = 2
$$ = 7709
$! =
$? = 0
$@ = 3 4
```

① width, length เป็นการประกาศตัวแปรสำหรับเอาไว้ในการคำนวณ

② นำค่าที่ส่งเข้ามาขณะทำการรันโปรแกรม ค่าที่ส่งเข้ามาจะถูกเก็บไว้ในตัวแปร built in $1 และ $2 จากนั้นนำค่าที่รับเข้ามาคูณกันแล้วเก็บไว้ในตัวแปรชื่อว่า result

③ ทดสอบพิมพ์ค่า built in ที่เก็บชื่อของเชลล์สคริปต์ที่กำลังทำงาน คือ $0 มีค่าเท่ากับ ./calRect

④ ทดสอบพิมพ์ค่า built in ที่เก็บค่าอาร์กิวเมนต์ที่ส่งเข้ามากับโปรแกรม ในตำแหน่งแรก คือ $1 มีค่าเท่ากับ 3

⑤ ทดสอบพิมพ์ค่า built in ที่เก็บค่าอาร์กิวเมนต์ที่ส่งเข้ามากับโปรแกรม ในตำแหน่งสอง คือ $2

⑥ ทดสอบพิมพ์ค่า built in ที่เก็บค่าอาร์กิวเมนต์ของโปรแกรมทั้งหมด คือ $* มีค่าเท่ากับ 3 4

⑦ ทดสอบพิมพ์ค่า built in ที่เก็บจำนวนของอาร์กิวเมนต์ทั้งหมด คือ $# มีค่าเท่ากับ 2

⑧ ทดสอบพิมพ์ค่า built in ที่เก็บอาร์กิวเมนต์ทั้งหมด คือ $$ มีค่าเท่ากับ 3 4

⑨ ทดสอบพิมพ์ค่า built in ที่เก็บค่าเก็บค่าตัวเลขโพรเซส (PID) ของโปรแกรมที่รันอยู่เบื้องหลังล่าสุด คือ $! ในที่นี้จะไม่มีค่า เพราะไม่ได้สั่งให้โปรแกรมทำงานเป็นแบบเบื้องหลัง

⑩ ทดสอบพิมพ์ค่า built in ที่เก็บค่าที่ถูกส่งคืนมาจากโปรแกรม คือ $? มีค่าเท่ากับ 0 หมายถึงโปรแกรมทำงานได้ปกติ

①① เหมือนกับ $*

ตัวแปรระบบชนิดสภาพแวดล้อม (System Environment Variables)

ตัวแปรสภาพแวดล้อมจริงๆแล้วก็คือตัวแปรธรรมดา แต่แตกต่างที่ตัวแปรประเภทดังกล่าวนี้จะมีการสืบทอดค่าของตัวแปรให้เชลล์อื่นๆ ที่กำลังทำงานด้วย (คล้ายกับการประกาศตัวแปรแบบโกล์บอล : global) ตัวอย่าง ตัวแปรสภาพแวดล้อมที่เราคุ้นเคยกันเช่น PATH, SHELL เป็นต้น การสร้างตัวแปรสภาพแวดล้อมเหมือนกับการสร้างตัวแปรทั่วไป แต่แตกต่างกันที่ตัวแปรสภาพแวดล้อม ต้องใช้คำสั่ง export ในการประกาศตัวแปรให้ระบบทราบ ตัวแปรสภาพแวดล้อมควรจะใช้ชื่อตัวแปรเป็นตัวอักษรตัวใหญ่ ตัวอย่างเช่น

```
# ENV_VAR="test environment variable"
# export ENV_VAR
# echo $ENV_VAR
test environment variable
```

หรือจะตั้งแค่ตัวแปรและประกาศตัวแปรสภาพแวดล้อมพร้อมๆกันก็ได้

```
# export ENV_VAR="test environment variable"
```

ทดสอบผลการประกาศตัวแปรสภาพแวดล้อม โดยการเขียนโปรแกรมเชลล์สคริปต์ที่แสดงค่าตัวแปรตามตัวอย่างต่อไปนี้ โดยตั้งชื่อว่า test_env

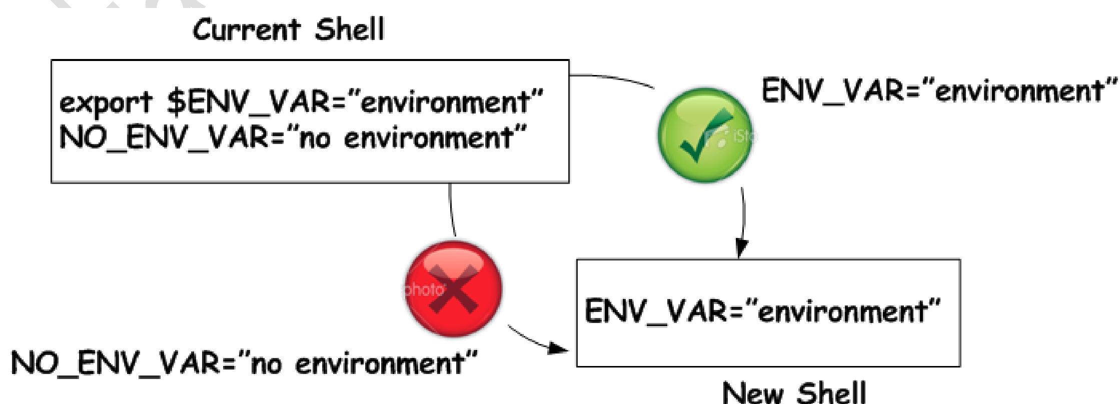
```
#!/bin/bash
# file name: test_env
# description: testing export environment variable
echo $ENV_VAR
echo $NO_ENV_VAR
```

ก่อนการทดสอบโปรแกรมให้ทำการกำหนดค่าให้กับตัวแปร ENV_VAR = “environment variable” เป็นชนิด environment และประกาศตัวแปร NO_ENV_VAR= “no environment” เป็นชนิดธรรมดา ดังตัวอย่าง

```
# export $ENV_VAR="environment"
# NO_ENV_VAR="no environment"
# ./test.env ①
environment
# . test.env ②
environment
no environment
```

① ทำการรันเชลล์สคริปต์โดยการเรียกเชลล์ตัวใหม่ (สร้างโปรเซสใหม่) เชลล์ตัวใหม่ในที่นี้คือบรรทัด #!/bin/bash นั่นเอง เชลล์ที่รันใหม่จะได้รับการสืบทอดตัวแปรสภาพแวดล้อม ENV_VAR มาด้วยทำให้สามารถแสดงผลได้ แต่ไม่สามารถแสดงค่าของตัวแปร NO_ENV_VAR ได้ เพราะไม่ใช่ตัวแปรสภาพแวดล้อม (ไม่สืบทอดไปสู่โปรแกรมอื่น)

② การรันเชลล์สคริปต์แบบที่สองโดยใช้ "." (dot) คือการใช้เชลล์ที่ทำงานอยู่ในปัจจุบันอ่านไฟล์ test_env โดยไม่มีการสร้างเชลล์ใหม่ ดังนั้นจึงสามารถแสดงค่าของตัวแปรทั้งสองตัวได้ คำสั่งที่เหมือนกับ "." คือคำสั่ง source ใช้สำหรับอ่านข้อมูลเข้าไปทำงานในเชลล์ที่กำลังใช้งานอยู่ ตัวอย่างเช่นไฟล์ ~/.bashrc จะมีการตั้งค่าตัวแปรต่างๆไว้ สำหรับกำหนดค่าเริ่มต้นเมื่อเชลล์เริ่มทำงาน ถ้ามีการแก้ไขไฟล์นี้แล้ว และต้องการให้มีผลทันทีโดยไม่ต้องเริ่มเชลล์ใหม่ก็ใช้คำสั่ง source ~/.bashrc ซึ่งจะทำการอ่านข้อมูลในไฟล์ดังกล่าว เข้าไปทำงานในเชลล์ที่ทำงานอยู่ในปัจจุบันทันที



รูปที่ 7-2 แสดงการ export ตัวแปรสภาพแวดล้อม

การแสดงค่าตัวแปรสภาพแวดล้อม

ผู้ใช้งานสามารถแสดงค่าต่างๆ ของตัวแปรสภาพแวดล้อมโดยใช้คำสั่ง export หรือ printenv ก็ได้

```
# export
declare -x ENV_VAR="environment"
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/root"
declare -x HOSTNAME="localhost"
declare -x INPUTRC="/etc/inputrc"
declare -x LANG="en_US.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="root"
declare -x
LS_COLORS="no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:cd=40;33;0
1:or=01;05;37;41:mi=01;05;37;41:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=01;32:*.btm=0
1;32:*.bat=01;32:*.sh=01;32:*.csh=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*
.lzh=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;31:*.bz2=01;31:*.bz=01;31:*.tz=01;31:*.
rpm=01;31:*.cpio=01;31:*.jpg=01;35:*.gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.
png=01;35:*.tif=01;35:"
declare -x MAIL="/var/spool/mail/root"
declare -x MY_ENV_VAR="test environment var"
declare -x OLDPWD="/root"
declare -x
PATH="/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/
usr/sbin:/usr/bin:/root/bin"
declare -x PWD="/root/SCRIPT"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_ASKPASS="/usr/libexec/openssh/gnome-ssh-askpass"
declare -x SSH_CLIENT="192.168.1.2 2010 22"
declare -x SSH_CONNECTION="192.168.1.2 2010 192.168.1.3 22"
declare -x SSH_TTY="/dev/pts/1"
declare -x TERM="vt100"
declare -x USER="root"
```

ถ้าต้องการยกเลิกตัวแปรสภาพแวดล้อมที่ได้สร้างไว้ ให้ใช้คำสั่ง export –n ตัวอย่างเช่น

```
# export –n ENV_VAR
```

ตัวแปรระบบชนิดสภาพแวดล้อมอื่นๆ

ในระบบลีนุกซ์และยูนิกซ์จะมีตัวแปรสงวนที่ระบบปฏิบัติการจะนำไปใช้ประโยชน์โดยเฉพาะเท่านั้น แต่หากผู้ใช้ทราบความหมายของตัวแปรเหล่านั้น ก็สามารถแก้ไขเปลี่ยนแปลงค่าของตัวแปรดังกล่าวได้ ผู้ใช้สามารถตรวจสอบค่าของตัวแปรระบบได้โดยใช้คำสั่ง set

Environment Variables	คำอธิบาย
$HOSTNAME	ชื่อเครื่องที่ใช้งาน
$PWD	ไดเรกทอรีที่ทำงานอยู่ เช่น /root/Script
$OLDPWD	ไดเรกทอรีก่อนหน้าไดเรกทอรีปัจจุบัน เช่น /root
$RANDOM	ตัวเลขสุ่มระหว่าง 0 และ 32767
$HOME	โฮมไดเรกทอรี
$PS1	ตัวแปรเก็บค่าพรอมต์หลัก (primary prompt)
$PS2	ตัวแปรเก็บค่าพรอมต์รอง (secondary prompt) ค่าโดยปริยายจะเป็น ">"
$PATH	ระบุเส้นทางเมื่อต้องการเรียกโปรแกรมให้ทำงาน (path) ระบบจะทำการค้นหาตัวโปรแกรมนั้นจากรายชื่อของไดเรกทอรีที่ได้กำหนดไว้ในตัวแปรนี้
$MAIL	ตำแหน่งไฟล์และไดเรคทรอรีที่เก็บอีเมลล์
$LOGNAME	ชื่อทะเบียนผู้ใช้ระบบ (login name)
$TERM	ชนิดของจอ (ในที่นี้เป็น vt100)
$SHELL	เชลล์ที่ใช้งานในปัจจุบัน

นอกจากการใช้คำสั่ง "set" เพื่อแสดงรายชื่อและค่าของตัวแปรต่างๆออกมาแล้วเรายังสามารถใช้คำสั่ง "echo <ชื่อตัวแปร>" เพื่อแสดงค่าของตัวแปรดังกล่าวออกมาก็ได้เช่นกัน และการเปลี่ยนหรือกำหนดค่าให้กับตัวแปรก็สามารถทำได้เช่นเดียวกับตัวแปรเชลล์ปกติทั่วไป ดังตัวอย่าง

```
# echo $PS1 ①
[\u@\h \W]\$  ②
$ PS1="Admin:> "   ③
Admin:> ④
```

① ให้แสดงค่าของตัวแปร PS1 (พร้อมพต์ของระบบ)

② ระบบพิมพ์ตัวอักษร "[root@localhost ~]#" ซึ่งถูกใช้เป็นพร้อมพต์ออกมาให้

③ กำหนดค่าของพร้อมพต์ใหม่เป็น "Admin:> "

④ พร้อมพต์ของระบบถูกเปลี่ยนเป็น "Admin:> "

ตัวแปรแถวลำดับ (array variable)

ตัวแปรแถวลำดับ (array) สามารถประกาศตัวแปรได้ดังนี้

name[index]=value

name คือชื่อของอาเรย์

index คือตัวเลขจำนวนเต็มที่ใช้สำหรับอ้างอิงค่าในอาเรย์

value คือค่าที่ต้องการใส่ในอาเรย์

```
# array[0]=apple
# array[1]=orange
# echo ${array[0]} ${array[1]}
apple orange
```

จากตัวอย่าง ทำการประกาศตัวแปรชื่อว่า array โดยกำหนดค่าให้กับอาเรย์ช่องแรก array[0] เท่ากับ apple และช่องที่ 2 array[1] เท่ากับ orange จากนั้นทำการแสดงข้อมูลที่เก็บไว้ โดยใช้เครื่องหมาย {} ช่วยในการแสดงผล การกำหนดค่าให้กับอาเรย์สามารถทำครั้งเดียวพร้อมๆ กันได้ ดังนี้

name=(value$_1$ ... value$_n$)

ตัวอย่าง เช่น

```
# days=(sunday monday tuesday wednesday thursday friday saturday)
# echo ${array[2]} ${array[6]}
tuesday saturday
```

คำสั่งรับข้อมูลแผ่นแป้นพิมพ์

ในเชลล์สคริปต์จะมีคำสั่งรับข้อมูลเข้า คือ คำสั่ง read โดยจะรับข้อมูลผ่านทางแป้นพิมพ์ แล้วนำไปเก็บไว้ในตัวแปร เพื่อใช้งานต่อไป

โดยมีรูปแบบดังนี้

read ตัวแปร

ในการรับข้อมูลแต่ละครั้งจะรับได้ครั้งละ 1 ตัวเท่านั้น

ตัวอย่างการใช้คำสั่งรับข้อมูล

```
# read input
abc123
# echo $input
abc123
```

คำสั่งในการแสดงผลทางจอภาพ

ในเชลล์สคริปต์จะมีคำสั่งในการแสดงผลทางจอภาพ คือ คำสั่ง echo ซึ่งเราสามารถจะแสดงที่เป็นค่าคงที่หรือเป็นตัวแปรก็ได้

รูปแบบของคำสั่ง echo

echo ตัวแปร

echo ค่าคงที่

echo "ตัวแปร"

echo "ค่าคงที่"

ในคำสั่ง echo จะมี option ประกอบดังนี้

-n หมายถึง พิมพ์ข้อมูลเสร็จแล้วไม่ต้องขึ้นบรรทัดใหม่

-e หมายถึง อนุญาตให้ใช้เครื่องหมายพิเศษเศษได้

เครื่องหมายพิเศษ

\a หมายความว่า จะมีแสดงดังออกมาจากลำโพง

\b หมายความว่า จะพิมพ์เครื่องช่องว่างออกทางจอภาพ

\c หมายความว่า จะพิมพ์โดยไม่สนใจเครื่องหมายขึ้นบรรทัดใหม่

\f หมายความว่า พิมพ์เสร็จให้เลื่อนเคอร์เซอร์ไปต้นบรรทัด

\n หมายความว่า พิมพ์ข้อความเสร็จแล้วให้ขึ้นบรรทัดใหม่

\r หมายความว่า พิมพ์เสร็จให้ส่งเครื่องหมาย ENTER ด้วย

\t หมายความว่า ในการพิมพ์ให้มีการจัดแท็บก่อนพิมพ์ในแนวนอน

\v หมายความว่า ในการพิมพ์ให้มีการจัดแท็บก่อนพิมพ์ในแนวตั้ง

\\ หมายความว่า ให้พิมพ์เครื่องหมาย \ ออกทางจอภาพ

\nnn หมายความว่า ให้พิมพ์ตัวอักษรจากรหัสแอสกี้(ASCII) โดยที่ nnn อยู่ในรูปเลขฐานแปด

ตัวอย่างการใช้คำสั่ง echo ในรูปแบบต่าง ๆ

```
# echo This is String
# str="This is String"
# echo $str
# echo "This is String"
# echo "$str"
This is String
This is String
This is String
This is String
```

ตัวอย่างการใช้คำสั่ง echo ในรูปแบบที่มีการใช้ option

```
# echo -n This is String
# str="This is String"
# echo -n $str
# echo "This is String"
# echo "$str"
This is String This is String This is String
This is String
```

ตัวอย่างโปรแกรเชลล์สคริปต์รับข้อมูลและแสดงผลข้อมูล

ตั้งชื่อโปรแกรมว่า read_echo ป้อนคำสั่งดังต่อไปนี้

```
#!/bin/bash
# read and write shell script
#
echo -n "Please enter you name : "
read name
echo "Hello $name"
```

เพิ่มสิทธิ์ในการประมวลผลให้กับไฟล์ พร้อมสั่งสคริปต์ให้ทำงาน

```
# chmod +x read_echo
# ./read_echo
Please enter you name : suchart  <enter> ← ป้อนชื่อ
Hello suchart
```

โปรแกรมจะพิมพ์คำว่า Please enter name: ออกทางจอภาพและจะไม่ขึ้นบรรทัดใหม่ รับข้อมูลจากแป้นพิมพ์ แล้วเก็บไว้ในตัวแปร name โดยเคอร์เซอร์จะรอรับต่อจากเครื่องหมาย : โปรแกรมจะพิมพ์คำว่า Hello และตามด้วยข้อมูลที่เราป้อนเข้าไป ตัวอย่างเช่น ถ้าเราป้อนคำว่า suchart ก็จะได้ผลเป็น Hello suchart

การเปลี่ยนทิศข้อมูล (Redirection)

สัญลักษณ์ที่ใช้ในการเปลี่ยนทิศทางมี 3 ประเภทได้แก่

> ผลลัพธ์ที่เกิดจากคำสั่งจะส่งไปเก็บไว้ในไฟล์ และเป็นแบบเขียนทับข้อมูลเดิม

>> ผลลัพธ์ที่เกิดจากคำสั่งจะส่งไปเก็บไว้ในไฟล์ และเป็นแบบเขียนต่อจากข้อมูลเดิม

< เป็นข้อมูลที่ได้จากไฟล์ส่งไปให้คำสั่งทำงานแทนข้อมูลที่ได้จากคีย์บอร์ด

ตัวอย่างสคริปต์การเปลี่ยนทิศข้อมูล

บันทึกไฟล์ชื่อ redirec

```
#!/bin/bash
clear
echo "ls > file_list"
ls > file_list  ①
echo "ls -la >> file_list"
ls -la >> file_list  ②
echo "cat < file_list"
cat < file_list  ③
```

เพิ่มสิทธิ์ในการประมวลผลให้กับไฟล์ พร้อมสั่งสคริปต์ให้ทำงาน

```
# chmod +x redirect
# ./redirect
ls > file_list
ls -la >> file_list
cat < file_list
calRect
```

```
file_list
HelloWorld.bash
myscript
read_echo
redirect
test_env
total 36
drwxr-xr-x  2 root root 4096 Jan 23 01:44 .
drwxr-x--- 28 root root 4096 Jan 22 20:03 ..
-rwxr-xr-x  1 root root  466 Jan 22 22:30 calRect
-rw-r--r--  1 root root   71 Jan 23 01:44 file_list
-rwxr-xr-x  1 root root   84 Jan 22 20:28 HelloWorld.bash
-rwxr-xr-x  1 root root    0 Jan 22 20:03 myscript
-rwxr-xr-x  1 root root  111 Jan 23 01:18 read_echo
-rwxr-xr-x  1 root root  146 Jan 23 01:44 redirect
-rwxr-xr-x  1 root root  117 Jan 22 23:32 test_env
```

จากโปรแกรม ① ทำการสั่งแสดงรายการในไดเรคทรอรีปัจจุบัน (ls) ผลลัพธ์ที่ได้จากคำสั่งดังกล่าวส่งไปเก็บไว้ในไฟล์ชื่อ file_list ด้วย > ② ทำการสั่งแสดงรายการในไดเรคทรอรีปัจจุบันแบบละเอียด (ls -al) ผลลัพธ์ที่ได้จากคำสั่งดังกล่าวส่งไปเก็บไว้ในไฟล์ชื่อ file_list เช่นเดิมแต่เป็นแบบต่อท้ายของไฟล์ ด้วย >> ③ ผลลัพธ์ที่เก็บไว้ใน file_list จะถูกเปลี่ยนทิศทางอีกครั้ง คือส่งไปให้ยังคำสั่ง cat เพื่อแสดงผลข้อมูลที่อยู่ในไฟล์ดังกล่าวออกทางจอภาพ

ข้อมูลมาตรฐาน

ข้อมูลมาตรฐานในเชลล์จะมีอยู่ 4 ชนิด คือข้อมูลเข้า(stdin), ข้อมูลแสดงผล(stdout), ข้อมูลข้อผิดพลาด(stderr), และแฟ้มข้อมูล(file) ในทางปฏิบัติ เราสามารถเปลี่ยนทิศทางของข้อมูลเหล่านี้ไปมาได้ โดยมีมาตรฐานคือ 1 จะหมายถึงข้อมูลแสดงผล(stdout) และ 2 จะหมายถึงข้อมูลความผิดพลาด(stderr)

ตัวอย่างเปลี่ยน stdout ไปเป็น file

```
# ls -l > ls-l.txt
```

จะเปลี่ยนการแสดงผลของคำสั่ง ls -l ไปเก็บไว้ที่ไฟล์ชื่อ ls-l.txt ดังนั้นคำสั่งตามตัวอย่างนี้จะไม่แสดงอะไรออกมาทางจอภาพ แต่จะเก็บไว้ที่ไฟล์แทน หากเราต้องการดูผล สามารถใช้คำสั่งแสดงผลของไฟล์ได้คือ

```
# cat ls-l.txt
```

ตัวอย่างเปลี่ยน stderr ไปเป็น file

```
# grep in * 2> errors.txt
```

ตัวอย่างนี้เป็นการค้นหาข้อความ in ในทุกไฟล์ (*) และหากเกิดข้อผิดพลาดขึ้น จะนำข้อความผิดพลาดไปเก็บไว้ที่ไฟล์ชื่อ errors.txt

ตัวอย่างเปลี่ยน stdout ไปเป็น stderr

```
# $ grep in * 1>&2
```

เป็นการค้นหาข้อความ da ในทุกไฟล์ (*) โดยนำการแสดงผลไปใส่ไว้ใน stderr แทนการแสดงผลปกติ แต่ในกรณีนี้เราป้อนคำสั่งทางแป้นพิมพ์ stdout และ stderr คือจอภาพเหมือนกัน จึงไม่เห็นความแตกต่าง แต่หากคำสั่งนี้ไปอยู่ในสคริปต์ที่เรากำหนดให้ stderr เป็นไฟล์ error-log การแสดงผลก็จะถูกเปลี่ยนทิศไปตามนั้น

ตัวอย่างเปลี่ยน stderr ไปเป็น stdout

```
# grep in * 2>&1
```

เป็นการค้นหาข้อความ in ในทุกไฟล์ (*) โดยหากเกิดข้อผิดพลาดขึ้น จะแสดงผลข้อผิดพลาดออกมาทาง stdout ซึ่งในที่นี้คือจอภาพเหมือนกัน

ตัวอย่างเปลี่ยน stderr และ stdout ไปยัง file

```
# rm -f $(find /home/USER -name somsak) &> /dev/null
```

คำสั่งนี้เป็นการค้นหาไฟล์ในไดเรกทอรี /home/USER ที่มีชื่อว่า somsak (find /home/USER -name somsak) เมื่อพบแล้วก็จัดการลบทิ้งโดยไม่เตือน (rm -f) โดยโยกการแสดงผลทั้งหมด (ทั้ง stderr และ stdout - ใช้สัญญลักษณ์ &>) ไปยังไฟล์ชื่อ /dev/null ซึ่งเป็นไฟล์พิเศษ หมายความว่ายกเลิกการแสดงผลทั้งหมด หรือคล้ายเป็นถังขยะของระบบ



คำสั่งนี้ค่อนข้างอันตราย เพราะลบโดยไม่เตือน โปรดทดลองด้วยความระมัดระวัง

การส่งต่อผลลัพธ์ หรือ ไปป์ (Pipes)

ไปป์เป็นการส่งต่อผลลัพธ์จากคำสั่งหนึ่งไปเป็นค่านำเข้าของอีกคำสั่งหนึ่ง

ตัวอย่างไปป์ 1

```
# ls -l | sed -e "s/[aeio]/u/g"
```

ตัวอย่างนี้จะนำเอาผลลัพธ์ที่ได้จากคำสั่ง ls -l ส่งต่อไปให้คำสั่ง sed -e "s/[aeio]/u/g" ซึ่งจะแปลงการแสดงผลจากอักขระ a หรือ e หรือ i หรือ o ไปเป็นอักขระ u ทั้งหมด

ตัวอย่างไปป์ 2

```
# ls -l | grep "\.txt$"
```

ตัวอย่างนี้จะส่งผลลัพธ์จากคำสั่ง ls -l ต่อไปให้คำสั่ง grep "\.txt$" คือให้แสดงเฉพาะไฟล์ที่มีนามสกุลเป็น .txt เท่านั้น มีค่าเท่ากับคำสั่ง ls แบบใส่พารามิเตอร์กรอง

 รูปแบบ "\.txt$" เป็นรูปแบบของ Regular Expression ซึ่งใช้มากในเชลล์สคริปต์ มีความหมายว่า "ที่ต้องลงท้ายด้วย .txt"

การตรวจสอบเงื่อนไขโดยการใช้คำสั่ง test

ผู้ใช้สามารถทำการตรวจสอบเงื่อนไขในเชลล์ได้ โดยการใช้คำสั่ง "test" โดยที่ถ้าเงื่อนไขที่ทำการตรวจสอบนั้นเป็นเงื่อนไขที่ถูกต้องโปรแกรม test จะส่งค่า return code เป็น 0 แต่ถ้าเป็นเงื่อนไขที่ไม่ถูกต้องจะให้ค่าที่ไม่เป็น 0 (โดยปกติจะเป็นค่า 1)

การตรวจสอบเงื่อนไขที่เกี่ยวกับไฟล์

คำสั่ง	คำอธิบาย
-d pathname	เป็นจริง ถ้ามีไฟล์อยู่และไฟล์เป็นแบบไดเรกทอรี
-e pathname	เป็นจริง ถ้ามีไฟล์อยู่จริง
-h pathname	เป็นจริง ถ้าเป็นไฟล์ชนิด Symbolic link (มีสัญลักษณ์ l นำหน้าไฟล์)
-f pathname	เป็นจริง ถ้ามีไฟล์อยู่และไฟล์เป็นแบบไฟล์ธรรมดา (ordinary)
-p pathname	เป็นจริง ถ้าเป็นไฟล์ชนิด Named pipe (มีสัญลักษณ์ p นำหน้าไฟล์)
-r pathname	เป็นจริง ถ้ามีไฟล์อยู่และอนุญาตให้สามารถอ่านไฟล์ได้ (readable)
-s pathname	เป็นจริง ถ้ามีไฟล์อยู่และเป็นไฟล์ที่มีข้อมูล ขนาดไฟล์ไม่เท่ากับ 0
-S pathname	เป็นจริง ถ้าเป็นไฟล์ชนิด Socket (มีสัญลักษณ์ s นำหน้าไฟล์)
-w pathname	เป็นจริง ถ้ามีไฟล์อยู่และอนุญาตให้สามารถเขียนไฟล์ได้ (writable)
-x pathname	เป็นจริง ถ้ามีไฟล์อยู่และอนุญาตให้สามารถทำงานได้ (executable)

การตรวจสอบเงื่อนไขที่เกี่ยวกับสตริง

คำสั่ง	คำอธิบาย
str1 = str2	เป็นจริง ถ้ามีไฟล์อยู่และไฟล์เป็นแบบไดเรกทอรี
str1 != str2	เป็นจริง ถ้า str1 ไม่เหมือนกับ str2
str	เป็นจริง ถ้าสตริงมีข้อมูลอยู่ (ไม่เป็น null)
-n str	เป็นจริง ถ้าสตริงมีความยาวไม่เป็นศูนย์
-z str	เป็นจริง ถ้าสตริงมีความยาวเป็นศูนย์

การตรวจสอบเงื่อนไขที่เกี่ยวกับตัวเลข

คำสั่ง	คำอธิบาย

x -eq y	เป็นจริง ถ้า x เท่ากับ y
x -ge y	เป็นจริง ถ้า x มากกว่าหรือเท่ากับ y
x -gt y	เป็นจริง ถ้า x มากกว่า y
x -le y	เป็นจริง ถ้า x น้อยกว่าหรือเท่ากับ y
x -lt y	เป็นจริง ถ้า x น้อยกว่า y
x -ne y	เป็นจริง ถ้า x ไม่เท่ากับ y

การรวมเงื่อนไข

คำสั่ง	คำอธิบาย
!	เปลี่ยนผลลัพธ์เงื่อนไขเป็นตรงกันข้าม
-o	เงื่อนไขแบบหรือ (OR)
-a	เงื่อนไขแบบและ (AND)
(...)	รวมคำสั่ง test

ตารางผลลัพธ์คณิตศาสตร์ลอจิกของ A กับ B

A	B	A and B (A –a B)	A or B (A –o B)
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

รูปแบบการใช้คำสั่ง test

การใช้คำสั่ง test สามารถแบ่งเป็นได้ 4 รูปแบบคือ

syntax	คำอธิบาย
test *condition*	ทดสอบเงื่อนไขใน contition ถ้า condition เป็นจริง ผลลัพธ์จะส่งไปยังตัวแปร $? มีค่าเป็น 0 แต่ถ้าผลลัพธ์เป็นเท็จ ตัวแปร $? จะมีค่าเป็น 1 เช่น test 3 -gt 5 (3 มากกว่า 5 เป็นเท็จ) เมื่อ echo $? จะเท่ากับ 1
test *condition && true-command*	ทดสอบเงื่อนไขใน contition ถ้า condition เป็นจริง จะทำงานต่อหลังจากเครื่องหมาย && ถ้าเป็นเท็จจะไม่แสดงผลลัพธ์ออกมา(แต่ยังคงเก็บไว้ในตัวแปร $? เช่นเดิม) เช่น

	test 5 -gt 3 && echo “5 > 3” จากตัวอย่าง 5 มากกว่า 3 จริงจึงพิมพ์ข้อความต่อท้าย && เป็น 5 > 3
test ***condition || false-command***	ทดสอบเงื่อนไขใน contition ถ้า condition เป็นเท็จ จะทำงานต่อหลังจากเครื่องหมาย || ถ้าเป็นจริงจะไม่แสดงผลลัพธ์ออกมา(แต่ยังคงเก็บไว้ในตัวแปร $? เช่นเดิม) เช่น test 3 -gt 5 && echo “false” จากตัวอย่าง 3 มากกว่า 5 เป็นเท็จ จึงพิมพ์ข้อความต่อท้าย || เป็น false
test ***condition && true-command || false-command***	ทดสอบเงื่อนไขใน contition ถ้า condition เป็นจริงจะทำคำสั่งหลังเครื่องหมาย && แล้วจบการทำงาน แต่ถ้า condition เป็นเท็จ จะคำสั่งต่อหลังจากเครื่องหมาย || เช่น test 3 -gt 4 && echo True || echo false จากตัวอย่าง 3 มากกว่า 4 เป็นเท็จ จึงพิมพ์ข้อความต่อท้าย || เป็น false

ในการใช้คำสั่ง test กับเงื่อนไข เมื่อคำสั่ง test ตรวจสอบเงื่อนไขนั้นว่าถูกต้องก็จะทำการคืนค่า exit code 0 ให้ จากหัวข้อที่แล้ว เราได้เรียนรู้ว่าหากต้องการจะตรวจสอบค่า exit code ของโปรแกรมหรือเชลล์เราสามารถตรวจสอบจากตัวแปร "$?" ได้ ซึ่งในที่นี้เราก็จะทำการพิมพ์ค่า exit code นั้นออกมาโดยการใช้คำสั่ง echo ตัวอย่างเช่น จะทดสอบว่า "/bin" เป็นไดเรกทอรีหรือไม่ ซึ่งก็จะได้ exit code จากคำสั่ง test เป็น 0 ซึ่งก็คือ "/bin" เป็นไดเรกทอรีจริง

```
# test -d /bin; echo $?
0
```

ตัวอย่างถัดไปจะทำการตรวจสอบว่า "/bin" เป็นไฟล์หรือไม่ ซึ่งผลลัพธ์ที่ได้ก็จะเป็นเท็จ

```
# test -f /bin; echo $?
1
```

ตัวอย่างถัดมาจะใช้เครื่องหมายก้ามปู "[...]" ทำการครอบเงื่อนไขที่ต้องการจะทดสอบซึ่งก็จะให้ผลลัพธ์แบบเดียวกับการใช้คำสั่ง test แต่การใช้เครื่องหมายก้ามปูจะทำให้บรรทัดคำสั่งดูเป็นระเบียบและเข้าใจง่ายกว่า

```
# [ ! -f /bin ]; echo $?
0
```

ตัวอย่างนี้จะใช้นิพจน์ทางตรรกะเข้ามาร่วมตรวจสอบเงื่อนไขด้วย คือเป็นการทดสอบว่าค่าตัวเลข 10 มากกว่า 15 และ ค่าตัวเลข 20 มากกว่า 19 หรือไม่ พิจารณาจากตารางทางตรรกะศาสตร์แล้ว ก็จะเห็นว่าผลลัพธ์ของ "เท็จ" และ "เท็จ" (F AND T) ก็จะได้ผลลัพธ์ออกมาเป็นเท็จ (F)

```
# [ 10 -gt 15 -a 20 -gt 19 ]; echo $?
1
```

ตัวอย่างนี้จะทำการตรวจสอบสอบข้อความว่าเหมือนกันหรือไม่ โดยใช้คำสั่ง ! ถ้าไม่เหมือนจะแสดงผลเป็นจริง

```
# [ "abc" != "def" ];echo $?
0
```

ตัวอย่างต่อไป จะทดสอบว่าค่าที่อยู่ใน $HOME เป็นไดเรคทรอรีหรือไม่ ผลคือจริง

```
# test -d "$HOME" ;echo $?
0
```

ตัวอย่างต่อไป จะทดสอบว่าค่า 5 มากกว่า 2 โดยใช้เครื่องหมาย > ร่วมด้วย

```
# test 5 > 2 && echo "Yes"
# test 1 > 2 && echo "Yes"
Yes
yes
```

ตัวอย่างต่อไป จะทดสอบคำสั่ง test กับเงื่อนไขหลายๆ แบบ

```
# test 5 = 5 && echo Yes || echo No
# test 5 = 15 && echo Yes || echo No
# test 5 != 10 && echo Yes || echo No
# test -f /etc/resolv.conf && echo "File /etc/resolv.conf found." || echo "File
/etc/resolv.conf not found."
# test -f /etc/resolv1.conf && echo "File /etc/resolv.conf found." || echo "File
/etc/resolv.conf not found."
Yes
No
Yes
File /etc/resolv.conf found.
File /etc/resolv.conf not found.
```

ตัวอย่างการใช้ test กับ Mathematical Operators (สร้างไฟล์สคริปต์ชื่อ testMath)

```
#!/bin/bash

if test $1 -gt 0
then
	echo "$1 > 0"
fi

if test $1 -ge 0
then
	echo "$1 >= 0"
fi
```

```
if test $1 -eq 0
then
	echo "$1 == 0"
fi

if test $1 -ne 0
then
	echo "$1 != 0"
fi

if test $1 -lt 0
then
	echo "$1 < 0"
fi

if test $1 -le 0
then
	echo "$1 <= 0"
fi
```

ผลการทำงาน

```
# chmod +x testMath
# ./testMath 0
0 >= 0
0 == 0
0 <= 0
```

ตัวอย่างการใช้ test กับ Logical Operators (สร้างไฟล์สคริปต์ชื่อ testLogic)

```
if test $1 -lt 0
then
	echo "$1 < 0"
fi

if test $1 -le 0
then
	echo "$1 <= 0"
fi
```

ผลการทำงาน

```
# chmod +x testLogic
# ./testLogic 0
0 <= 0
```

ตัวอย่างการใช้ test กับ String Operators (สร้างไฟล์สคริปต์ชื่อ testString)

```
string_null=""
string1="string1"

if [ $string_null -n ]
```

```
then
        echo "not null string"
else
        echo "null string"
fi

if [ $string_null -z ]
then
        echo "null string"
else
        echo "not null string"
fi

if [ "$string_null" == "$string1" ]
then
        echo "strings equal"
else
        echo "strings not equal"
fi

if [ "$string_null" != "$string1" ]
then
        echo "strings not equal"
else
        echo "strings equal"
fi
```

ผลการทำงาน

```
# chmod +x testString
# ./testString
not null string
null string
strings not equal
strings not equal
```


การใช้คำสั่งควบคุมทิศทางการทำงาน (Flow control command)

ผลลัพธ์ที่ได้จากการใช้คำสั่ง test เราสามารถนำมาใช้เป็นเงื่อนไขของการควบคุมทิศทางการทำงานของโปรแกรมได้ ซึ่งก็จะขึ้นอยู่กับรูปแบบของคำสั่งที่ควบคุมทิศทางต่างๆว่าจะจัดการกับเงื่อนไขที่เป็นจริงอย่างไร และจะจัดการกับเงื่อนไขที่เป็นเท็จอย่างไร

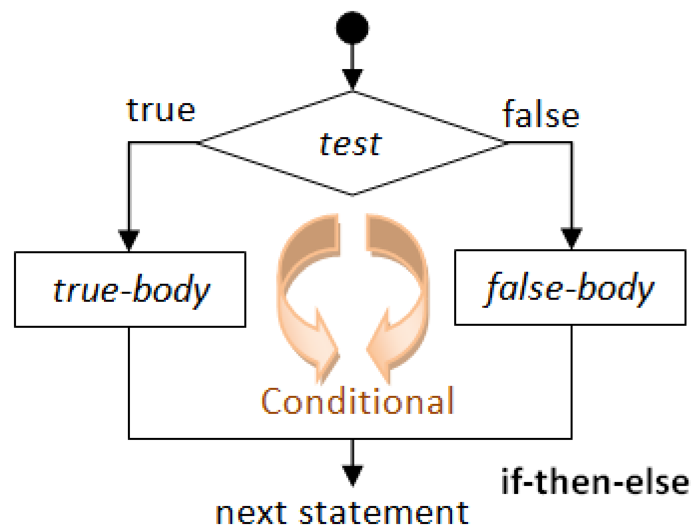
การควบคุมทิศทางการทำงานแบบ ถ้า...แล้ว (if...then)

รูปแบบของคำสั่ง

```
if
then
        command list
```

```
else
        command list
fi
```

Flow chart ของ if…then



สำหรับคำสั่งควบคุมทิศทางแบบ ถ้า...แล้ว นี้หากเงื่อนไขที่ทดสอบคืนค่า exit code ที่เป็นจริง (0) มาให้ก็จะทำคำสั่งหลัง "then" แต่ถ้าเป็นเท็จ (ไม่ 0) ก็จะทำชุดคำสั่งที่อยู่หลัง "else" คำสั่งควบคุมทิศทางแบบ ถ้า...แล้ว จะต้องทำการปิดชุดคำสั่งด้วยคำสำคัญ "fi" (เป็นตัวกลับหน้า-หลังกับ if) ตัวอย่าง ให้สร้างไฟล์ชื่อว่า ifelse1 ซึ่งมีคำสั่งดังนี้

```
if [ -d $1 ]
then
    echo $1 is a directory
else
    echo $1 is not a directory
fi
```

บันทึกไฟล์ จากนั้นกำหนดสิทธิ์ให้สามารถประมวลผลไฟล์ได้ ด้วยคำสั่ง chmod +x ifelse1 และสั่งให้สคริปต์ทำงานด้วยคำสั่ง ./ifelse1 /bin ผลลัพธ์ที่ได้จะแสดงข้อความว่า /bin is a directory ผลมาจากการใช้คำสั่ง test –d ตรวจสอบว่าเป็นไดเรคทรอรีหรือไม่ ซึ่งเป็นจริง จากนั้น if ตรวจสอบการทำงานต่อ ค่าที่ได้เป็นจริงจึงทำงานหลัง then ทันที ต่อจากนั้นให้ทำการทดสอบเปลี่ยนค่าอากิวเมนต์ที่ป้อนให้กับเชลล์เป็น /biz ผลที่ได้คือ /biz is not a directory ซึ่งเชลล์ทำการตรวจสอบไฟล์ดังกล่าวแล้วปรากฏว่าไม่มีในระบบจึงส่งค่ากลับเป็นเท็จ (จะไปทำงานหลัง else)

```
# chmod +x ifelse1
# ./ifelse1 /bin
/bin is a directory
# ./ifelse1 /biz
/biz is not a directory
```

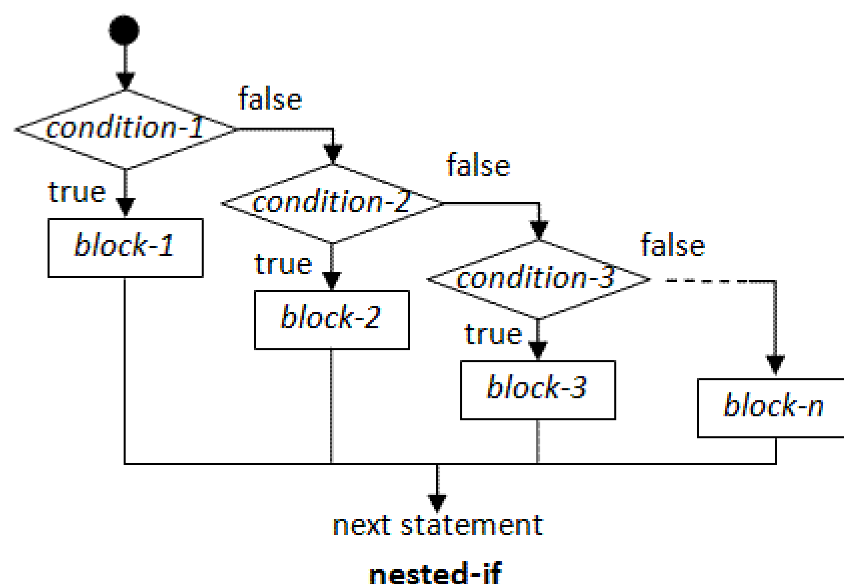
การควบคุมทิศทางการทำงานแบบ if...then…elif

สำหรับคำสั่งควบคุมทิศทางแบบ ถ้า...แล้ว รูปแบบที่สองนี้จะคล้ายกับแบบแรก เพียงแต่มีการเพิ่มเติมการตรวจเงื่อนไขเพิ่มขึ้นมาอีกเงื่อนไขหนึ่ง ซึ่งเงื่อนไขนี้จะอยู่ตามหลังคำบังคับ "elif" หากเงื่อนไขตรงนี้เป็นจริง ก็จะทำงานตามชุดคำสั่งหลังคำบังคับ "then" ที่ตามมาทันที สำหรับเงื่อนไขของชุดคำสั่งแบบนี้ สามารถจะเพิ่มเติมเข้ามาได้ไม่จำกัดจำนวน ในกรณีที่เงื่อนไขเกิดถูกต้องมากกว่าหนึ่งเงื่อนไข ก็จะเข้าไปทำงานชุดคำสั่งแรกตามเงื่อนไขแรกสุดที่พบว่าตรงกัน และจะไม่สนใจเงื่อนไขอื่นๆที่ตรงกันอีก หากเงื่อนไขทั้งหมดของ "if" และ "elif" ไม่มีเงื่อนไขใดที่เป็นจริงเลย ก็จะไปทำชุดคำสั่งหลังคำบังคับ "else" แทน

รูปแบบของคำสั่ง

```
if
then
        command list
elif
then
        command list
elif
then
        command list
…
else
        command list
fi
```

Flow chart ของ if…then…elif



nested-if

ตัวอย่าง ให้เขียนโปรแกมตามตัวอย่างและบันทึกเป็น ifelif

```
if [ "$1" = "Apple" ]
then
   echo "My fruit is $1"
```

```
elif [ "$1" = "Orange" ]
then
    echo "My fruit is $1"
else
    echo "I don't know $1"
fi
```

ทดลองรันโปรแกรมดังนี้

```
# chmod +x ifelif
# ./ifelif Apple
My fruit is Apple
# ./ifelif Orange
My fruit is Orange
# ./ifelif Banana
I don't know Banana
```

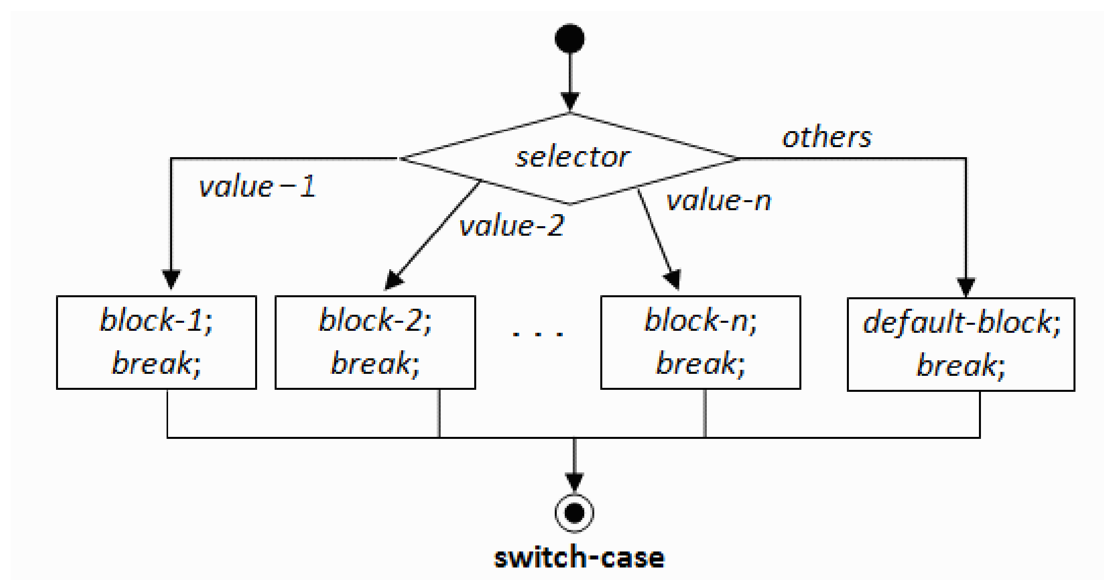
คำสั่งควบคุมทิศทางแบบ case

หากมีเงื่อนไขแบบ "ถ้า...แล้ว" หรือ if-elif จำนวนมาก เราจะมีวิธีที่ดีกว่าในการจัดการกับเงื่อนไขเหล่านั้น นั่นก็คือใช้ case เข้ามาแทนที่ สำหรับโครงสร้างของ case ตามปกติจะใช้จับคู่ (matching) กับสตริงที่มีรูปแบบลักษณะ (pattern) เหมือนกันและจะไม่สามารถใช้ได้กับรูปแบบของนิพจน์ทางคณิตศาสตร์ สำหรับรูปแบบต่างๆจะต้องมีเครื่องหมายวงเล็บปิด ")" ตามท้าย และหลังจากนั้นก็จะเป็นชุดคำสั่งที่จะให้ทำงานจะต้องปิดท้ายชุดคำสั่งด้วยเครื่องหมาย colon คู่ ";;" เราสามารถใส่รูปแบบได้มากกว่าหนึ่งรูปแบบได้ สำหรับรูปแบบที่ไม่สามารถจับคู่ได้เลย จะใช้คำบังคับ "*" การปิดท้ายคำสั่งควบคุม case จะใช้คำบังคับ "esac" (เป็นคำกลับหน้า-หลังของ "case")

รูปแบบของคำสั่ง

```
case STRING in
    pattern1)  command-list1 ;;
    pattern2)  command-list2 ;;
    pattern3)  command-list3 ;;
           *)  cammand-list-else ;;
esac
```

Flow chart ของ case



ตัวอย่าง case (บันทึกเป็น case_1)

```
echo -n "Are you continue process ? [y/Y/n/N]"; read INPUT
case "$INPUT" in
    [yY]) echo "Your job are processing !" ;;
      [nN]) echo "Your job are cencled" ;;
         *) echo "entry is false"
            echo "the valid entry are [y/Y/n/N]" ;;
esac
```

สำหรับตัวอย่างการใช้ case ข้างบนจะเป็นการแสดงถึงการจับคู่ของตัวอักษร ซึ่งเราจะอ่านเข้ามาเก็บไว้ในตัวแปรเชลล์ชื่อ INPUT และจะนำเอาค่าในตัวแปร INPUT ไปเปรียบเทียบแบบสตริงกับรูปแบบต่างๆ การใช้คำสั่ง echo -n จะเป็นการพิมพ์ข้อความที่ตามหลัง echo แบบไม่ขึ้นบรรทัดใหม่ ดังนั้นเมื่อใช้คำสั่ง read ตามมาก็จะเป็นการรอรับการใส่ข้อมูลต่อท้ายจากข้อความที่พิมพ์ออกมาทันที (ไม่ขึ้นบรรทัดใหม่) จากตัวอย่างเราจะเห็นว่ารูปแบบของสตริงแต่ละบรรทัดสามารถกำหนดให้จับคู่กันได้มากกว่าหนึ่งแบบ ตัวอย่างเช่น "[yY]" ก็จะเป็นการเลือกได้ว่าจะจับคู่กับตัวอักษร "y", "Y" เนื่องจากอยู่ในเครื่องหมายก้ามปู ซึ่งมีความหมายว่าสามารถเลือกเอาได้หนึ่งอย่างจากที่อยู่ในเครื่องหมายก้ามปูนั้น ส่วนบรรทัดสุดท้ายเป็นการจับคู่กับสตริงที่ไม่เข้าคู่เลยนั้นจะเห็นว่าเราสามารถใส่ชุดคำสั่งเข้าไปมากกว่าหนึ่งบรรทัดได้ แต่ชุดคำสั่งสุดท้ายจะต้องปิดท้ายด้วยเครื่องหมาย colon คู่ ";;"

ทดลองรันโปรแกรมดังนี้

```
# chmod +x case_1
# ./case_1
Are you continue process ? [y/Y/n/N]y
Your job are processing !

# ./case_1
Are you continue process ? [y/Y/n/N]n
```

```
Your job are cencled

# ./case_1
Are you continue process ? [y/Y/n/N]d
entry is false
the valid entry are [y/Y/n/N]
```

ตัวอย่างที่ 2 (บันทึกเป็น case_2)

```
# if no vehicle name is given
# i.e. -z $1 is defined and it is NULL
#
# if no command line arg

if [ -z $1 ]
then
        rental="*** Unknown vehicle ***"
elif [ -n $1 ]
then
# otherwise make first arg as rental
        rental=$1
fi

case $rental in
   "car") echo "For $rental 20 $ per day";;
   "van") echo "For $rental 10 $ per day";;
   "jeep") echo "For $rental 5 $ per day";;
   "bicycle") echo "For $rental 20 paisa per day";;
   *) echo "Sorry, I can not gat a $rental for you";;
esac
```

สำหรับตัวอย่างการใช้ case ตัวอย่างที่ 2 ข้างบนจะเป็นการตรวจสอบราคาเช่ารถยนต์ชนิดต่างๆ การทำงานเริ่มจาก รับค่าอากิวเมนต์มาจากคีย์บอร์ด(ชนิดของรถยนต์ที่ต้องการเช่า) แล้วทำการตรวจสอบว่าข้อมูลดังกล่าวไม่เป็น null ด้วยคำสั่ง "[-z $1]" ถ้าเป็น null จะทำการกำหนดค่า rental เท่ากับ Unknown vehicle ถ้าไปเป็น null จะกำหนดค่าของ rental เป็นชื่อรถยนต์ที่ต้องการ จากนั้นมาเข้า case อีกครั้งเพื่อเลือกว่ารถยนต์ที่ผู้เช่าเลือกนั้น จะคิดราคาเท่าไหร่ เช่น ถ้าผู้เช่าเลือกรถยนต์ชนิด van จะต้องจ่ายค่าเช่า 10 $ ต่อวัน ถ้ารถยนต์ที่เลือกไม่ตรงกับ case ใดๆ จะทำที่ *) คือ พิมพ์คำว่า Sorry, I can not gat a $rental for you ออกแทน

ทดลองรันโปรแกรมดังนี้

```
# chmod +x case_2
# ./case_2 van
For van 10 $ per day
# ./case_2 truck
Sorry, I can not gat a truck for you
```

คำสั่งควบคุมทิศทางแบบวนรอบ (loop flow control)

คำสั่งควบคุมทิศทางแบบวนรอบหรือวนลูปจะมีอยู่สามแบบคือ for, while และ until

for loop

รูปแบบของคำสั่ง

```
for var in list
do
        commands
done
```

คำสั่ง for นี้จะทำทำงานวนรอบจนกระทั่งสมาชิกใน list ถูกใช้หมด ชุดคำสั่งในลูป for จะต้องอยู่ภายใต้คำบังคับ "do" และ "done" การใช้ for ส่วนใหญ่จะใช้กับลูปที่รู้จำนวนของรอบที่ต้องทำ

ตัวอย่าง for (บันทึกเป็น for_1)

```
for i in 1 2 3 4 5
do
        echo "Welcome $i times"
done
```

ให้บันทึกเชลล์สคริปต์ข้างต้นเป็นไฟล์ชื่อ for_1 แล้วสั่งให้โปรแกรมทำงานดังนี้

```
# chmod +x for_1
# ./for_1
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
```

จากตัวอย่างเป็นการวนลูบแสดงผลเลข 1 – 5 ออกทางจอภาพ

ตัวอย่างต่อไป เป็นการแสดงผลรวมตัวเลขจำนวนเต็มตั้งแต่ 1 ถึง 9 เสร็จแล้วพิมพ์ค่าที่ได้ออกมา (บันทึกไฟล์สคริปต์ชื่อ for_sum)

```
SUM=0
for i in 1 2 3 4 5 6 7 8 9
do
        SUM=`expr $SUM + $i`
done
        echo "Sum is $SUM"
```

ผลลัพธ์

```
# chmod +x for_sum
# ./for_sum
Sum is 45
```

ตัวอย่างต่อไป เป็นการรับค่าจากคีย์บอร์ด โดยเริ่มต้นตรวจสอบว่าอากิวเมนต์ที่ป้อนเข้ามาถูกต้องหรือไม่ “[$# -eq 0]” เมื่อค่าที่ป้อนเข้ามาผิดพลาดจะออกจากโปรแกรมแบบทำงานไม่สมบูรณ์

"exit 1" ถ้าข้อมูลที่ป้อนเข้ามาถูกต้องก็จะทำการคูณกับจำนวนครั้งที่วนรอบไปเรื่อยจนกว่าจะครบ 10 ครั้ง ด้วยคำสั่ง "$n * $i = `expr $i \* $n`"

ให้บันทึกสคริปต์ชื่อว่า for_mul

```
#!/bin/sh
#
#Script to test for loop
#
#
if [ $# -eq 0 ]
then
        echo "Error - Number missing form command line argument"
        echo "Syntax : $0 number"
        echo "Use to print multiplication table for given number"
        exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10 #or for ((  i =     0 ;  i <= 10;  i++  ))
do
        echo "$n * $i = `expr $i \* $n`"
done
```

ผลลัพธ์

```
# chmod +x for_mul
# ./for_mul 2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

คำสั่งควบคุมทิศทางแบบ while และ until

รูปแบบของคำสั่งของ while

```
while (true condition)
do
    command list
done
```

คำสั่ง while จะใช้กับเงื่อนไขที่เป็นจริง

ตัวอย่าง (บันทึกชื่อเป็น while_1)

```
SUM=0
i=1
while [ $i -lt 10 ]
do
    SUM=`expr $SUM + $i`
    i=`expr $i + 1`
done
      echo "Sum is $SUM"
```

ผลลัพธ์

```
# chmod +x while_1
# ./while_1
Sum is 45
```

โปรแกรมตัวอย่างข้างบนจะทำงานเหมือนกับตัวอย่างของ for แต่ในที่นี้ใช้รูปแบบของ while loop โปรแกรมจะทำการบวกตัวเลข "i" ไปเรื่อยๆ โดยเริ่มตั้งแต่ 1 และเพิ่มค่า i ขึ้นทีละหนึ่ง ลูปนี้จะทำงานไปเรื่อยๆตราบใดที่ค่า i ยังน้อยกว่า 10 เมื่อค่า i มากกว่า 10 แล้วก็จะออกจากลูปและพิมพ์ผลบวกที่ได้ทั้งหมด

ตัวอย่างที่ 2 เป็นการคูณเหมือน for แต่ใช้ while แทน (บันทึกชื่อเป็น while_2)

```
#!/bin/sh
#
#Script to test while statement
#
if [ $# -eq 0 ]
then
   echo "Error - Number missing form command line argument"
   echo "Syntax : $0 number"
   echo " Use to print multiplication table for given number"
exit 1
fi
n=$1
i=1
while [ $i -le 10 ]
do
  echo "$n * $i = `expr $i \* $n`"
  i=`expr $i + 1`
done
```

ผลลัพธ์

```
# chmod +x while_2
# ./while_2 2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
```

```
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

รูปแบบของคำสั่งของ until

```
until (false condition)
do
    command list
done
```

คำสั่ง until จะใช้กับเงื่อนไขที่เป็นเท็จ

ตัวอย่าง (บันทึกชื่อเป็น until_1)

```
SUM=0
i=1
until [ $i -eq 10 ]
do
    SUM=`expr $SUM + $i`
    i=`expr $i + 1`
done
        echo "Sum is $SUM"
```

โปรแกรมตัวอย่างข้างบนก็เป็นโปรแกรมที่ให้ผลลัพธ์แบบเดียวกับโปรแกรมที่ใช้โครงสร้างของ for-loop และ while-loop เช่นเดียวกัน จะเห็นว่าโปรแกรมมีลักษณะที่คล้ายกับโปรแกรมที่ใช้โครงสร้างแบบ while-loop มาก เพียงแต่เงื่อนไขที่ใช้ตรวจสอบจะเป็นเงื่อนไขที่กลับกันเท่านั้น กล่าวคือเงื่อนไขนี้จะใช้ตรวจสอบว่าค่าของตัวแปร i เท่ากับ 10 หรือไม่ until-loop จะทำการวนลูปทำงานไปเรื่อยๆตราบใดที่เงื่อนไขที่ตามหลัง until มานั้นยังคงเป็นเท็จ ในการเขียนโปรแกรมนั้นเราสามารถจะเลือกใช้เฉพาะ while-loop หรือ until-loop ตัวใดตัวหนึ่งเพียงอย่างเดียวก็ได้ เพราะเงื่อนไขของลูปทั้งสองแบบเราสามารถดัดแปลงให้ทดแทนกันได้

break และ continue

คำสั่ง break และ continue จะทำการหยุดการทำชุดคำสั่งที่ตามหลัง break กับ continue และสำหรับ break เมื่อหยุดแล้วก็จะกระโดดออกไปนอกลูป ตรงจุดที่อยู่ถัดจากคำบังคับ "done" ส่วน continue จะกลับไปทำงานในลูปต่อ ตรงจุดที่อยู่ถัดจากคำบังคับ "do"

```
SUM=0
while [ true ]
do
    echo -n "Please enter number 1-9 , except 5 , (q-quit)"
    read INPUT
```

```
    if [ "$INPUT" = "q" ]
    then
        echo "Break loop now!"
        break
    elif [ $INPUT -eq 5 ]
    then
        echo "Please not input 5"
        continue
    elif [ $INPUT -lt 1 ]
    then
        echo "Please not input data < 1"
        continue
    elif [ $INPUT -gt 9 ]
    then
        echo "Please do not input data > 9"
        continue
    fi
    SUM=`expr $SUM + $INPUT`
done
    echo "SUM is $SUM"
```

ผลลัพธ์

```
# chmod +x break_con
# ./break_con
Please enter number 1-9 , except 5 , (q-quit)1
Please enter number 1-9 , except 5 , (q-quit)3
Please enter number 1-9 , except 5 , (q-quit)5
Please not input 5
Please enter number 1-9 , except 5 , (q-quit)7
Please enter number 1-9 , except 5 , (q-quit)9
Please enter number 1-9 , except 5 , (q-quit)q
Break loop now!
SUM is 20
```

ในตัวอย่างนี้นอกจากแสดงถึงการใช้ break กับ continue แล้ว ยังแสดงให้เห็นถึงการใช้ while ที่เป็นที่นิยมใช้อีกแบบหนึ่ง นั่นก็คือการใช้กับเงื่อนไข "true" วิธีการใช้งานแบบนี้ จะทำให้โปรแกรมวนลูปแบบไม่รู้จบ ซึ่งมักจะใช้กับการวนทำงานที่มีการรับข้อมูลเข้าไปตามตัวอย่างข้างต้น จะมีเงื่อนไขของการรับข้อมูลบางรูปแบบที่จะใช้กระโดดออกจากลูปเพื่อจบการทำงาน ซึ่งในที่นี้ก็จะใช้คำสั่ง "break" เมื่อผู้ใช้มีการป้อนตัวอักษร "q" เข้ามา เราอาจจะใช้คำสั่ง "exit" แทนก็ได้ แต่โปรแกรมจะไม่ทำงานส่วนที่พิมพ์ผลลัพธ์ที่อยู่ท้ายโปรแกรม สำหรับคำสั่ง "continue" จะใช้กับการป้อนข้อมูลที่เป็นตัวเลข 5 ซึ่งในที่นี้ก็จะเป็นการข้ามการบวกเลข 5 ไป ตัวเลขที่ใส่ได้และโปรแกรมจะทำการบวกเลขให้จะมีเฉพาะตัวเลข 1-4 และ 6-9 ส่วนตัวเลขที่น้อยกว่า 1 และมากกว่า 9 ก็จะถูกคำสั่ง "continue" ข้ามส่วนของโปรแกรมที่เป็นการคำนวณการบวกเลขไปเช่นเดียวกัน คำสั่ง continue และ break นี้จะทำการกระโดดออกไปจากลูปที่โปรแกรมกำลังทำงานอยู่ แต่เรา

สามารถระบุให้ continue และ break ทำการกระโดดออกไปยังลูปที่ชั้นใดก็ได้ ในกรณีที่ลูปของเรามีการครอบอยู่หลายชั้น วิธีการใช้งานก็เพียงแต่เติมตัวเลขตามท้ายคำสั่ง break และ continue เท่านั้น ตัวอย่างเช่น "break 2" จะทำการกระโดดออกจากลูปไปสองชั้น (ไปอยู่หลัง done ของลูปที่สอง)

ฟังก์ชัน

เราสามารถสร้างฟังก์ชันในเชลล์ด้วยคำสั่ง function มีรูปแบบดังนี้

รูปแบบของคำสั่งของ function

```
function name {
   command
   ...
}
```

คำที่อยู่หลัง function เป็นชื่อของฟังก์ชันที่ตั้งขึ้นเอง ฟังก์ชันสามารถรับอาร์กิวเมนต์ได้ โดยอ้างอิงตัวแปรตำแหน่ง $1, $2, $3 ... หมายถึงอาร์กิวเมนต์ตัวที่ 1, 2, 3

ตัวอย่างต่อไปนี้เป็นการสร้างฟังก์ชันหาผลลัพธ์ของผลบวก ตั้งแต่ 1 ถึงจำนวนเต็มที่ต้องการ โดยให้จำนวนเต็มที่ต้องการเป็นอาร์กิวเมนต์ของฟังก์ชัน (ตั้งชื่อเป็น func_1)

```
#!/bin/sh

function sum {
    s=0
    for i in `seq 1 $1`
        do
        s=$(($s+$i))
    done
    echo $s
}

sum 55
```

ผลลัพธ์

```
# chmod +x func_1
# ./func_1
1540
```

หลังจากที่นิยามฟังก์ชันแล้วก็สามารถใช้ชื่อฟังก์ชันนั้น เป็นคำสั่งเหมือนคำสั่งอื่นๆ ในความเป็นจริงแล้วเราสามารถละเว้นคำว่า function จะไม่เขียนก็ได้ สมมติว่าไฟล์นี้ชื่อ func_1, เมื่อรันแล้วจะได้ผลลัพธ์เป็น 1540

ตัวอย่างฟังก์ชันแบบที่ 2

ตัวอย่างนี้ จะทำการสร้างฟังก์ชัน 2 ฟังก์ชันคือ quit และ hello สำหรับหน้าที่การทำงานของฟังก์ชัน hello คือ แสดงคำว่า Hello! ส่วนฟังก์ชันการทำงานของ quit คือออกจากโปรแกรม สำหรับคำสั่ง echo Hi จะไม่ถูกทำงานเนื่องจากสคริปต์จะทำงานจบที่คำสั่ง quit (บันทึกไฟล์ชื่อว่า func_2)

```
#!/bin/bash
function quit {
    exit
}
function hello {
    echo Hello!
}
hello
quit
echo Hi
```

ผลลัพธ์

```
# chmod +x func_2
# ./func_2
Hello!
```

การส่งผ่านค่าตัวแปรให้กับฟังก์ชัน

การส่งผ่านค่าตัวแปรเข้าไปในฟังก์ชั่นจะอาศัย ตัวแปร $1 ถ้ามีการส่งผ่านตัวแปรหลายตัว ก็จะใช้ตัวแปรที่เหลือคือ $2, $3, ...,$9 ดังตัวอย่าง (บันทึกไฟล์ชื่อว่า func_3)

```
#!/bin/bash
function quit {
    exit
}
function print {
    echo $1
}
print Hello
print World
quit
```

ผลลัพธ์

```
# chmod +x func_3
# ./func_3
Hello
World
```

บทสรุป

ในบทนี้ได้กล่าวถึงการเขียนโปรแกรมเชลล์สคริปต์เบื้องต้น โดยเริ่มต้นจากการประกาศตัวแปร การควบคุมทิศทางของโปรแกรมแบบต่างๆ เช่น if, if-then, case, while และจบท้ายด้วยการเขียน

ฟังก์ชันและการส่งค่าตัวแปรในฟังก์ชัน เนื้อหาทั้งหมดเป็นเพียงส่วนหนึ่งของการเขียนโปรแกรมเชลล์สคริปต์ เท่านั้น ยังมีเนื้อหาส่วนอื่นๆ อีก ซึ่งผู้อ่านสามารถอ่านเพิ่มเติมได้จากเอกสารอ้างอิงที่แนะนำไว้ท้ายหนังสือเล่มนี้

แบบฝึกหัดท้ายบท

1. จงสร้างสคริปต์สำหรับดูรายชื่อไฟล์ในไดเรคทอรี่ย่อย
2. จงสร้างสคริปต์บีบอัดสำรองข้อมูล
3. จงสร้างสคริปต์เปลี่ยนชื่อไฟล์ทีละหลายไฟล์
4. How to write shell script that will add two nos, which are supplied as command line argument, and if this two nos are not given show error and its usage
5. Write Script to find out biggest number from given three nos. Nos are supplies as command line argument. Print error if sufficient arguments are not supplied.
6. Write script to print nos as 5,4,3,2,1 using while loop.
7. Write Script, using case statement to perform basic math operation as follows

   + addition

   - subtraction

   x multiplication

   / division

   The name of script must be 'q4' which works as follows

   $ ./q4 20 / 3, Also check for sufficient command line arguments
8. Write Script to see current date, time, username, and current directory
9. Write script to print given number in reverse order, for eg. If no is 123 it must print as 321.
10. Write script to print given numbers sum of all digit, For eg. If no is 123 it's sum of all digit will be 1+2+3 = 6.
11. How to perform real number calculation in shell script and store result to third variable , lets say a=5.66, b=8.67, c=a+b?
12. Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument
13. Write script to determine whether given command line argument ($1) contains "*" symbol or not, if $1 does not contains "*" symbol add it to $1, otherwise show message

"Symbol is not required". For e.g. If we called this script Q12 then after giving ,

$ Q12 /bin

Here $1 is /bin, it should check whether "*" symbol is present or not if not it should print Required i.e. /bin/*, and if symbol present then Symbol is not required must be printed.

Test your script as

$ Q12 /bin

$ Q12 /bin/*

14. Write script to print contains of file from given line number to next given number of lines. For e.g. If we called this script as Q13 and run as

    $ Q13 5 5 myf , Here print contains of 'myf' file from line number 5 to next 5 line of that file.

15. Write script to implement getopts statement, your script should understand following command line argument called this script Q14,

    Q14 -c -d -m -e

    Where options work as

    -c clear the screen

    -d show list of files in current working directory

    -m start mc (midnight commander shell) , if installed

    -e { editor } start this { editor } if installed

16. Write script called sayHello, put this script into your startup file called .bash_profile, the script should run as soon as you logon to system, and it print any one of the following message in infobox using dialog utility, if installed in your system, If dialog utility is not installed then use echo statement to print message : -

    Good Morning

    Good Afternoon

    Good Evening , according to system time.

17. How to write script, that will print, Message "Hello World" , in Bold and Blink effect, and in different colors like red, brown etc using echo command.

18. Write script to implement background process that will continually print current time in upper right corner of the screen , while user can do his/her normal job at $ prompt.

19. Write shell script to implement menus using dialog utility. Menu-items and action according to select menu-item is as follows:

Menu-Item	Purpose	Action for Menu-Item
Date/time	To see current date time	Date and time must be shown using infobox of dialog utility
Calendar	To see current calendar	Calendar must be shown using infobox of dialog utility
Delete	To delete selected file	First ask user name of directory where all files are present, if no name of directory given assumes current directory, then show all files only of that directory, Files must be shown on screen using menus of dialog utility, let the user select the file, then ask the confirmation to user whether he/she wants to delete selected file, if answer is yes then delete the file , report errors if any while deleting file to user.
Exit	To Exit this shell script	Exit/Stops the menu driven program i.e. this script

Note: Create function for all action for e.g. To show date/time on screen create function show_datetime().

20. Write shell script to show various system configuration like

1) Currently logged user and his logname

2) Your current shell

3) Your home directory

4) Your operating system type

5) Your current path setting

6) Your current working directory

7) Show Currently logged number of users

8) About your os and version ,release number , kernel version

9) Show all available shells

10) Show mouse settings

11) Show computer cpu information like processor type, speed etc

12) Show memory information

13) Show hard disk information like size of hard-disk, cache memory, model etc

14) File system (Mounted)

21. Write shell script using for loop to print the following patterns on screen

<table>
<tr><th>for2</th><th>for3</th><th>for4</th></tr>
<tr><td>1
22
333
4444
55555</td><td>1
12
123
1234
12345</td><td>|_
| |_
| | |_
| | | |_
| | | | |_</td></tr>
<tr><th>for5</th><th>for6</th><th>for7</th></tr>
<tr><td>*
* *
* * *
* * * *
* * * * *</td><td>*
* *
* * *
* * * *
* * * * *
* * * * *
* * * *
* * *
* *
*</td><td></td></tr>
<tr><th>for8</th><th>for8</th><th>for9</th></tr>
<tr><td>1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9</td><td></td><td></td></tr>
</table>

เอกสารอ้างอิง

[1] http://en.wikipedia.org/wiki/Unix

[2] http://www.computerhope.com/history/unix.htm

[3] http://th.wikipedia.org/wiki/ยูนิกซ์

[4] http://th.wikipedia.org/wiki/ลินุกซ์

[5] Linux Online, Inc.; **List of distributions**; http://www.linux.org/dist/list.html; 11 january 2010.

[6] Wikipedia, the free encyclopedia; **List of Linux distributions**; http://en.wikipedia.org/wiki/List_of_Linux_distributions; 10 January 2010.

[7] DistroWatch; **DistroWatch Weekly, Issue 336**; http://distrowatch.com/; 2010.

[8] http://tutorials.beginners.co.uk/the-structure-of-unix.htm

[9] http://th.wikipedia.org/wiki/เอ็กซ์วินโดวซิสเต็ม

[10] http://th.wikipedia.org/wiki/บูรพาลีนุกซ์

[11] http://www.comptechdoc.org/os/linux/usersguide/linux_ugfilestruct.html

[12] http://en.wikipedia.org/wiki/Ext3

[13] http://en.wikipedia.org/wiki/Ext4

[14] http://en.wikipedia.org/wiki/Unix_file_types

[15] Linux **file system layou**; http://www.linux.co.uk

[16] Linux Directory Tree; Learning Debian GNU/Linux; http://oreilly.com/catalog/debian/chapter/book/appa_01.html

[17] ดุจใจ เรืองเวหา; **หน่วยเก็บข้อมูลสำรอง (Secondary Storage Unit)**; http://cptd.chandra.ac.th/selfstud/it4life/sub%20hard5.htm

[18] Microsoft Corporation.; **Disk Devices and Partitions**; http://msdn.microsoft.com/en-us/library/aa363966%28VS.85%29.aspx

[19] **ฮาร์ดดิสก์ (Hard disk) อุปกรณ์คอมพิวเตอร์**; http://www.modify.in.th/Hardware-Computer/Hard-disk-id32.aspx

[20] Wikimedia Foundation, Inc.; **Master boot record**; http://en.wikipedia.org/wiki/Master_boot_record; 21 December 2009

[21] M. Tim Jones, Consultant Engineer, Emulex; **Inside the Linux boot process**; http://www.ibm.com/developerworks/linux/library/l-linuxboot/; january 9 2010.

[22] Anthony Lissot; **Linux Partition HOWTO**; http://tldp.org/HOWTO/Partition/; January 9 2010.

[23] ภูวดล ด่านระหาญ; **System Bootup and Shutdown Process**; http://www.thaicert.org/paper/unix_linux/bootup_shutdown.php; january 9 2010,

[24] YoLinux.com ; **Linux Init Process / PC Boot Procedure**; http://www.yolinux.com/TUTORIALS/LinuxTutorialInitProcess.html; january 9 2010.

[25] Wikimedia Foundation, Inc.; **Unix shell**; http://en.wikipedia.org/wiki/Unix_shell; 4 January 2010

[26] Wikimedia Foundation, Inc.; **Bash shell**; http://en.wikipedia.org/wiki/Bash; 5 January 2010

[27] LinuxGuide.it; **Linux Command Line in English**; http://www.linuxguide.it/linux_commands_line_en.htm; 9 january 2010.

[28] David Tarsi; **Linux Command Reference**; http://www.perpetualpc.net/srtd_commands_rev.html; 5 january 2009.

[29] Computer Hope (tm); **Brief overview of Unix / Linux commands**; http://www.computerhope.com/unix/overview.htm; january 2010.

[30] O'Reilly Media, Inc.; **Linux Command Directory**; http://oreilly.com/linux/command-directory/; january 2010.

[31] O'Reilly Media, Inc.; **Linux in a Nutshell, 5th Edition**; http://www.oreillynet.com/linux/cmd/; january 2010.

[32] O’Reilly & Associates, Inc.; **Linux Quick Reference**; www.oreilly.com; fabuary 2010.

[33] FOSSWire; **Unix/Linux Command Reference**; www.fosswire.com; fabuary 2010.

[34] International Business Machines Corp(IBM); **Commands Reference, Volume 5**; http://publib.boulder.ibm.com; fabuary 2010.

[35] Wikipedia, the free encyclopedia; **Domain Name System**; http://en.wikipedia.org/wiki/Domain_Name_System; march 4, 2010;

[36] Laurent Gregoire; **about Vim**; http://www.vim.org/about.php; 2007

[37] By Ellie Quigley; **UNIX® Shells by Example**, Third Edition; Prentice Hall PTR; October 01, 2001

[38] Mark G. Sobell; **A Practical Guide to Linux® Commands**, Editors, and Shell Programming; Prentice Hall PTR; July 01, 2005

[39] Chris F.A. Jhhnson; **Pro Bash Programming: Scripting the GNU/Linux Shell**; Apress.

[40] Peter Seebach; **Beginning Portable Shell Scripting From Novice to Professional**; Apress.

[41] Graham Glass, King Ables; **Linux for Programmers and Users**; Prentice Hall; February 15, 2006

[42] Cameron Newham, Bill Rosenblatt; **Learning the bash Shell**, Second Edition; O'Reilly; January 1998

[43] Nelson H.F. Beebe, Arnold Robbins; **Classic Shell Scripting**; O'Reilly; May 2005

[44] Stephen G. Kochan, Patrick Wood; **Unix® Shell Programming**, Third Edition; Sams Publishing; February 27, 2003

[45] Dave Taylor ; **Wicked Cool Shell Scripts: 101 Scripts for Linux**, Mac OS X, and Unix Systems; Starch Press © 2004

เฉลยบทที่ 5

1. arch, arch –k
2. cal
3. cal 4 2009
4. cal 2012
5. cal -3m
6. cat /proc/cpuinfo, cat /proc/interrupts, cat /proc/meminfo, cat /proc/swaps, cat /proc/version, cat /proc/net/dev, cat /proc/mounts
7. clock
8.

1	date '+DATE: %m/%d/%y%nTIME:%H:%M:%S'
2	date +"Hello%t Date is %D %n%t Time is %T"

9. date -s "19 Apr 2007 09:35:10"
10. dmidecode --type 1
11. hdparm
12. lspci
13. lsusb
14. uname –a, uname –all
15. init 6
16. init 3
17. exit, logout, ctrl + D
18. reboot
19. shutdown +15 “Server is going down !!!”
20. shitdown –h now
21. shutdown +3 –r
22. cd /var/log
23. No such file or directory
24. cd ~
25. ls –al /etc/init.d/
26. ls –F /usr/bin
27. ls –t /etc
28. cp /root/install.log /home
29. cp –Rf /var/log /usr/local/bklog

30. cp *.txt /tmp
31. find / -name “inst*”
32. find /usr/local/ -name '*all*'
33. find /usr/ -name '*Doc*' -exec rm {} \;
34. find /usr/local/ -mtime -2
35. find /home -user test
36. find . -perm -644
37. find / -size +15000k
38. find /usr/ -mmin -5 -name '*.java'
39. find . \! -name '[A-Z] *' -exec echo >> foo.txt { } \;
40. find . -regex './t[0-9]'
41. ln –s /etc/init.d/sshd lnsshd
42. mkdir work; cd work; mkdir mgr net acc; cd net; mkdir it cs;
43. mkdir –m 222 person
44. mkdir –p work/it/network/os
45. mv *.* /tmp or mv * /tmp
46. mv pic1.jpg pic2.jpg
47. mv old_dir new_dir
48. mv –i file1 file2
49. pwd
50. rm –r /test
51. rm –i myfile
52. rm –i file1 file2 file3
53. touch test1 test2
54. touch –am test1
55. touch -d '1 Jan 2010 11:30' test2
56. touch -t 201509140810.50 test1
57. tree
58. tree –L 3
59. tree -dCA -L 3 /etc
60. locate perl
61. locate –i install
62. whereis find
63. which ssh
64. ueradd Harry -u 1005 -g users -c "Harry Potter" -d /home/Harry
65. userdel somsri

66. userdel –r somsri
67. groupadd NGROUP
68. groupadd IT_GROUP –g 550
69. groupmod –g 451 ENG
70. groupmod –o –g 451 IT
71. grpck /etc/gshadow
72. passwd tom
73. passed –d tom
74. passwd –x 90 tom
75. chmod 666 chfile.txt
76. chmod 751 file
77. chmod -R 644 documents
78. chown sutida data.txt
79. chown –R sutida install
80. chown –R test:users documents
81. chattr +a /log/test_log, lsattr /log/test_log
82. chattr +i test.conf, lsattr test.conf
83. tar –czvf /backup/backup01.tar.gz /home/*
84. tar –xzvf /backup/backup01.tar.gz
85. tar -cvzf /backup/myfile.tar.gz /home/myfile.txt
86. tar -cjvf BK1.tbz a.txt d.txt e.txt
87. bzip2 a.txt b.txt
88. bunzip2 a.bz2
89. tar -cjf allfile.tar.bz2 file1 file2 file3
90. gzip myfile
91. gzip -r dir1
92. cat /etc/passwd
93. cat -n file1.txt
94. cat >> file1.txt
95. more /var/log/messages
96. more –c /var/log/messages
97. head file1.txt
98. head -20 file1.txt, head –n 20 file1.txt
99. head –c1k file1.txt
100. less file.txt
101. tail myfile

102. tail myfile.txt -n 20
103. tail -20b bigfile
104. tail –f /var/log/message.log
105. echo Hello World, echo 'Hello World', echo "Hello World
106. echo –e "Warning !!! \a"
107.

```
echo '***********************************'
echo '*            MAIN MENU            *'
echo '***********************************'
echo '*  1. Computer Programming        *'
echo '*  2. Database Design             *'
echo '*  3. Operating System            *'
echo '*  4. Exit                        *'
echo '***********************************'
```

108.

```
sort file1 > filesort1; sort file2 > filesort2
comm filesort1 filesort2
```

109. diff -w myfile1 myfile2
110. grep mail /etc/passwd
111. grep –n mail /etc/passwd
112. grep –c nologin /etc/passwd
113. grep -r "Connect:127.0.0.1" /etc/
114. cat /proc/cpuinfo | grep cpu
115. grep –color tty /etc/group
116. ls | paste - - -
117. paste –d "=" f1.txt f2.txt, paste -s -d + f1.txt f2.txt
118. sed 's/scripts/Domain Name Server/g' file1.txt
119. sed -e 's/string1//g' file1.txt
120. sed -n '/string1/p' file1.txt
121. sort file.txt
122. ls -al | sort -n -k5
123. nl myfile.txt
124. od myfile.txt
125. od --address-radix=x /bin/sh

126. stat myfile
127. wc file1
128. wc -ml /etc/passwd
129. du /etc
130. du –sh /etc
131. file *
132. whereis ls
133. whereis –bm ls
134. cut -d: -f1,5,7 /etc/passwd
135. cut -c1-5 /etc/passwd
136. cut -c1-5,7-15 /etc/passwd
137. paste file1 file2 > file3
138. paste -s -d* file1 file2
139. ls | paste - - -
140. cat columns.txt | tr '[a-z]' '[A-Z]' > UpCaseColumns.txt
141. uniq –d myfileduplicate
142. date | tee file1 | less
143. cmp file1 file2
144. cmp –s file1 fil2 && echo "no diff"
145. md5sum /etc/passwd > myPasswd.md5
146. md5sum -c myPasswd.md5
147. df
148. df –Tah
149. mkdir /usr/logs; mount /var/log /usr/logs
150. mount /dev/fd0 /mnt/floppy
151. mount /dev/sda1 /mnt/floppy (สมมุติว่า flash drive อยู่ที่ /dev/sda1)
152. mount /dev/cdrom /mnt/cdrom
153. fsck
154. fsck /dev/hd1
155. dump -0f nklog /var/log
156. restore -if bklog
157. rsync -v -e ssh /www/backup.tar.gz suchart@archive.msu.ac.th:~
158. rsync -v -e ssh suchart@archive.msu.ac.th:~/ backup.tar.gz /tmp
159. lpr /etc/passwd
160. ps –f
161. ps –aux

162. w
163. top
164. top –u httpd
165. free –sk 3
166. kill –s SIGTERM 1234, kill –SIGHUP 1234
167. N/A
168. nice -5 vi myfile.txt, nice -7 top &, nice --12 init
169. watch –n 3 free –m
170. watch –n 1 --differences ifconfig etho
171. at -f myjob.sh 00:00 หรือ at -f myjob.sh 0000 4/25/2010
172. at -f myjob.sh 1700 +3 month
173. */5 * * * * ls –l
174. 00 1 * * 0 <command>
175. 30 8 1 * * <command>
176. hostname
177. ifconfig eth0
178. ifconfig eth0 up, ifconfig eth0 down
179. ifconfig eth0 192.168.1.100 netmask 255.255.255.0
180. ifconfig eth0 hw ether 01:02:03:04:05:0A
181. host www.kernel.org
182. host –C www.kernel.org
183. whois www.google.com
184. ping www.google.co.th
185. traceroute www.kernel.org

เฉลยแบบฝึกหัดบทที่ 7

1.

```
#!/bin/bash
function listdir {
   local PAT="$1"
   local ROOT="$2"
   for i in *; do
      if [ -d "$i" ]; then
         local CUR="$ROOT/$i"
         pushd "$i" &>/dev/null
         listdir "$PAT" "$CUR"
         popd &>/dev/null
      fi
   done
   if [ ! -z "$( ls -d $PAT 2>/dev/null )" ]; then
      echo "Directory: $ROOT"
      ls -d $PAT 2>/dev/null
      echo
   fi
}

if [ -z "$1" ]; then
  echo List file in PATTERN recursive into directories.
  echo Usage: $0 "PATTERN"
  exit
fi
PATTERN="$1"
echo "List $PATTERN"
listdir "$PATTERN" "."
```

2.

```
#!/bin/bash
SRCD="/home/"
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

3.

```
#!/bin/sh
# renna: rename multiple files according to several rules
# written by felix hudson Jan - 2000

#first check for the various 'modes' that this program has
#if the first ($1) condition matches then we execute that portion of the
#program and then exit

# check for the prefix condition
if [ $1 = p ]; then

#we now get rid of the mode ($1) variable and prefix ($2)
  prefix=$2 ; shift ; shift

# a quick check to see if any files were given
# if none then its better not to do anything than rename some non-existent
# files!!

  if [$1 = ]; then
     echo "no files given"
     exit 0
  fi

# this for loop iterates through all of the files that we gave the program
# it does one rename per file given
  for file in $*
    do
    mv ${file} $prefix$file
  done

#we now exit the program
  exit 0
fi
```

```
# check for a suffix rename
# the rest of this part is virtually identical to the previous section
# please see those notes
if [ $1 = s ]; then
  suffix=$2 ; shift ; shift

   if [$1 = ]; then
    echo "no files given"
   exit 0
   fi

 for file in $*
  do
   mv ${file} $file$suffix
 done

 exit 0
fi

# check for the replacement rename
if [ $1 = r ]; then

  shift

# i included this bit as to not damage any files if the user does not specify
# anything to be done
# just a safety measure

  if [ $# -lt 3 ] ; then
    echo "usage: renna r [expression] [replacement] files... "
    exit 0
  fi

# remove other information
  OLD=$1 ; NEW=$2 ; shift ; shift
```

```
# this for loop iterates through all of the files that we give the program
# it does one rename per file given using the program 'sed'
# this is a sinple command line program that parses standard input and
# replaces a set expression with a give string
# here we pass it the file name ( as standard input) and replace the nessesary
# text

  for file in $*
  do
    new=`echo ${file} | sed s/${OLD}/${NEW}/g`
    mv ${file} $new
  done
exit 0
fi

# if we have reached here then nothing proper was passed to the program
# so we tell the user how to use it
echo "usage;"
echo " renna p [prefix] files.."
echo " renna s [suffix] files.."
echo " renna r [expression] [replacement] files.."
exit 0

# done!
```

4.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q1.Script to sum to nos
#
```

```
if [ $# -ne 2 ]
then
   echo "Usage - $0   x    y"
   echo "        Where x and y are two nos for which I will print sum"
   exit 1
fi
   echo "Sum of $1 and $2 is `expr $1 + $2`"
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

5.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q2. Script to find out bigest number
#
# Algo:
#     1) START: Take three nos as n1,n2,n3.
#     2) Is n1 is greater than n2 and n3, if yes
#        print n1 is bigest no goto step 5, otherwise goto next step
#     3) Is n2 is greater than n1 and n3, if yes
#        print n2 is bigest no goto step 5, otherwise goto next step
#     4) Is n3 is greater than n1 and n2, if yes
#        print n3 is bigest no goto step 5, otherwise goto next step
#     5) END
#
#


   if [ $# -ne 3 ]
```

```
    then
            echo "$0: number1 number2 number3 are not given" >&2
      exit 1
    fi
    n1=$1
    n2=$2
    n3=$3
    if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
    then
            echo "$n1 is Bigest number"
    elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
    then
            echo "$n2 is Bigest number"
    elif [ $n3 -gt $n1 ] && [ $n3 -gt $n2 ]
    then
      echo "$n3 is Bigest number"
    elif [ $1 -eq $2 ] && [ $1 -eq $3 ] && [ $2 -eq $3 ]
    then
            echo "All the three numbers are equal"
    else
      echo "I can not figure out which number is biger"
    fi

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

6.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
```

```
# Q3
# Algo:
#       1) START: set value of i to 5 (since we want to start from 5, if you
#          want to start from other value put that value)
#       2) Start While Loop
#       3) Chechk, Is value of i is zero, If yes goto step 5 else
#          continue with next step
#       4) print i, decement i by 1 (i.e. i=i-1 to goto zero) and
#          goto step 3
#       5) END
#
i=5
while test $i != 0
do
                echo "$i
"
                i=`expr $i - 1`
done
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

7.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q4
#

if test $# = 3
then
```

```
        case $2 in
         +) let z=$1+$3;;
         -) let z=$1-$3;;
         /) let z=$1/$3;;
         x|X) let z=$1*$3;;
         *) echo Warning - $2 invalied operator, only +,-,x,/ operator allowed
            exit;;
        esac
        echo Answer is $z
else
        echo "Usage - $0   value1  operator value2"
        echo "        Where, value1 and value2 are numeric values"
        echo "              operator can be +,-,/,x (For Multiplication)"
fi



#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

8.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q5
#
echo "Hello, $LOGNAME"
echo "Current date is `date`"
echo "User is `who i am`"
echo "Current direcotry `pwd`"
```

```
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

9.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Script to reverse given no
#
# Algo:
#       1) Input number n
#       2) Set rev=0, sd=0
#               3) Find single digit in sd as n % 10 it will give (left most digit)
#       4) Construct revrse no as rev * 10 + sd
#       5) Decrment n by 1
#       6) Is n is greater than zero, if yes goto step 3, otherwise next step
#       7) Print rev
#
if [ $# -ne 1 ]
then
   echo "Usage: $0   number"
   echo "       I will find reverse of given number"
   echo "       For eg. $0 123, I will print 321"
   exit 1
fi


n=$1
rev=0
sd=0
```

```
while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    rev=`expr $rev \* 10  + $sd`
    n=`expr $n / 10`
done
    echo  "Reverse number is $rev"


#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

10.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Algo:
#       1) Input number n
#       2) Set sum=0, sd=0
#               3) Find single digit in sd as n % 10 it will give (left most digit)
#       4) Construct sum no as sum=sum+sd
#       5) Decrment n by 1
#       6) Is n is greater than zero, if yes goto step 3, otherwise next step
#       7) Print sum
#
if [ $# -ne 1 ]
then
    echo "Usage: $0   number"
    echo "       I will find sum of all digit for given number"
    echo "       For eg. $0 123, I will print 6 as sum of all digit (1+2+3)"
    exit 1
```

```
fi


n=$1
sum=0
sd=0
while [ $n -gt 0 ]
do
   sd=`expr $n % 10`
   sum=`expr $sum + $sd`
   n=`expr $n / 10`
done
   echo  "Sum of digit for numner is $sum"



#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

11.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q10
#
a=5.66
b=8.67
c=`echo $a + $b | bc`
echo "$a + $b = $c"


#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
```

```
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

12.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q11

if [ $# -ne 1 ]
then
   echo "Usage - $0  file-name"
   exit 1
fi

if [ -f $1 ]
then
   echo "$1 file exist"
else
   echo "Sorry, $1 file does not exist"
fi



#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

13.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
```

```
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q12
# Script to check whether "/*" is included, in $1 or not
#

cat "$1" > /tmp/file.$$   2>/tmp/file0.$$

grep "*"  /tmp/file.$$    >/tmp/file0.$$

if [ $? -eq 1 ]
then
   echo "Required i.e. $1/*"
else
   echo "Symbol is Not required"
fi

rm -f /tmp/file.$$
rm -f /tmp/file0.$$
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

14.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q13
#
```

```
# Shell script to print contains of file from given line no to next
# given  numberlines
#


#
# Print error / diagnostic for user if no arg's given
#
if [ $# -eq 0 ]
then
   echo "$0:Error command arguments missing!"
   echo "Usage: $0 start_line   uptoline   filename"
   echo "Where start_line is line number from which you would like to print file"
   echo "uptoline is line number upto which would like to print"
   echo "For eg. $0 5 5 myfile"
   echo "Here from myfile total 5 lines printed starting from line no. 5 to"
   echo "line no 10."
   exit 1
fi


#
# Look for sufficent arg's
#


   if [ $# -eq 3 ]; then
               if [ -e $3 ]; then
                  tail +$1 $3 | head -n$2
       else
                  echo "$0: Error opening file $3"
                  exit 2
               fi
   else
      echo "Missing arguments!"
   fi


#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
```

```
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

15.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q14
# -c clear
# -d dir
# -m mc
# -e vi { editor }
#


#
# Function to clear the screen
#
cls()
{
   clear
   echo "Clear screen, press a key . . ."
   read
   return
}


#
# Function to show files in current directory
#
show_ls()
{
   ls
   echo "list files, press a key . . ."
```

```
   read
   return
}

#
# Function to start mc
#
start_mc()
{
   if which mc > /dev/null ; then
                mc
                echo "Midnight commander, Press a key . . ."
                read
   else
                echo "Error: Midnight commander not installed, Press a key . . ."
                read
   fi
   return
}

#
# Function to start editor
#
start_ed()
{
   ced=$1
   if which $ced > /dev/null ; then
                $ced
                echo "$ced, Press a key . . ."
                read
   else
                echo "Error: $ced is not installed or no such editor exist, Press a key . . ."
                read
   fi
   return
}
```

```
#
# Function to print help
#
print_help_uu()
{
        echo "Usage: $0 -c -d -m -v {editor name}";
        echo "Where -c clear the screen";
        echo "      -d show dir";
        echo "      -m start midnight commander shell";
        echo "      -e {editor}, start {editor} of your choice";
        return
}

#
# Main procedure start here
#
# Check for sufficent args
#

if [ $# -eq 0 ] ; then
   print_help_uu
   exit 1
fi

#
# Now parse command line arguments
#
while getopts cdme: opt
do
   case "$opt" in
        c) cls;;
        d) show_ls;;
        m) start_mc;;
        e) thised="$OPTARG"; start_ed $thised ;;
        \?) print_help_uu; exit 1;;
```

```
    esac
done





#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

16.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q15
#

temph=`date | cut -c12-13`
dat=`date +"%A %d in %B of %Y (%r)"`

if [ $temph -lt 12 ]
then
    mess="Good Morning $LOGNAME, Have nice day!"
fi

if [ $temph -gt 12 -a $temph -le 16 ]
then
    mess="Good Afternoon $LOGNAME"
fi

if [ $temph -gt 16 -a $temph -le 18 ]
then
```

```
   mess="Good Evening $LOGNAME"
fi

if which dialog > /dev/null
then
   dialog --backtitle "Linux Shell Script Tutorial"\
   --title "(-: Welcome to Linux :-)"\
   --infobox "\n$mess\nThis is $dat" 6 60
   echo -n "                    Press a key to continue. . .                  "
   read
   clear
else
   echo -e "$mess\nThis is $dat"
fi



#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

17.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q16
# echo command with escape sequance to give differnt effects
#
# Syntax: echo -e "escape-code your message, var1, var2 etc"
# For eg. echo -e "\033[1m  Hello World"
#               |       |
#               |       |
```

```
#           Escape code   Message
#

clear
echo -e "\033[1m Hello World"
 # bold effect
echo -e "\033[5m Blink"
      # blink effect
echo -e "\033[0m Hello World"
 # back to noraml

echo -e "\033[31m Hello World"
 # Red color
echo -e "\033[32m Hello World"
 # Green color
echo -e "\033[33m Hello World"
 # See remaing on screen
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
echo -e "\033[36m Hello World"

echo -e -n "\033[0m "
  # back to noraml

echo -e "\033[41m Hello World"
echo -e "\033[42m Hello World"
echo -e "\033[43m Hello World"
echo -e "\033[44m Hello World"
echo -e "\033[45m Hello World"
echo -e "\033[46m Hello World"



echo -e "\033[0m Hello World"
  # back to noraml
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
```

```
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

18.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q17
# To run type at $ promot as
# $ q17 &
#

echo
echo "Digital Clock for Linux"
echo "To stop this clock use command kill pid, see above for pid"
echo "Press a key to continue. . ."

while :
do
   ti=`date +"%r"`
   echo -e -n "\033[7s"    #save current screen postion & attributes
   #
   # Show the clock
   #

   tput cup 0 69         # row 0 and column 69 is used to show clock

   echo -n $ti           # put clock on screen

   echo -e -n "\033[8u"   #restore current screen postion & attributs
   #
   #Delay fro 1 second
```

```
  #
  sleep 1
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

19.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

show_datetime()
{
  dialog --backtitle "Linux Shell Tutorial" --title "System date and Time" --infobox "Date is `date`"
3 40
  read
  return
}

show_cal()
{
  cal > menuchoice.temp.$$
  dialog --backtitle "Linux Shell Tutorial" --title "Calender" --infobox "`cat menuchoice.temp.$$`" 9
25
```

```
  read
  rm -f menuchoice.temp.$$
  return
}

delete_file()
{
 dialog --backtitle "Linux Shell Tutorial" --title "Delete file"\
 --inputbox "Enter directory path (Enter for Current Directory)"\
 10 40  2>/tmp/dirip.$$
 rtval=$?

 case $rtval in
    1) rm -f /tmp/dirip.$$ ; return ;;
    255) rm -f /tmp/dirip.$$ ; return ;;
 esac

 mfile=`cat /tmp/dirip.$$`

 if [ -z $mfile ]
 then
    mfile=`pwd`/*
 else
    grep "*" /tmp/dirip.$$
    if [ $? -eq 1 ]
    then
            mfile=$mfile/*
    fi
 fi

 for i in $mfile
 do
   if [ -f $i ]
   then
            echo "$i Delete?" >> /tmp/finallist.$$
   fi
```

```
done



dialog --backtitle "Linux Shell Tutorial" --title "Select File to Delete"\
--menu "Use [Up][Down] to move, [Enter] to select file"\
20 60 12 `cat /tmp/finallist.$$` 2>/tmp/file2delete.tmp.$$

rtval=$?

file2erase=`cat /tmp/file2delete.tmp.$$`

case $rtval in
   0) dialog --backtitle "Linux Shell Tutorial" --title "Are you shur"\
    --yesno "\n\nDo you want to delete : $file2erase " 10 60

      if [ $? -eq 0 ] ; then
              rm -f  $file2erase
       if [ $? -eq 0 ] ; then
          dialog --backtitle "Linux Shell Tutorial"\
                 --title "Information: Delete Command" --infobox "File: $file2erase is Sucessfully
deleted,Press a key" 5 60
                 read
                else
                 dialog --backtitle "Linux Shell Tutorial"\
                 --title "Error: Delete Command" --infobox "Error deleting File: $file2erase, Press a
key" 5 60
          read
         fi
              else
               dialog --backtitle "Linux Shell Tutorial"\
               --title "Information: Delete Command" --infobox "File: $file2erase is not deleted,
Action is canceled, Press a key" 5 60
               read
              fi
   ;;
   1)  rm -f /tmp/dirip.$$ ; rm -f /tmp/finallist.$$ ;
```

```
        rm -f /tmp/file2delete.tmp.$$; return;;
    255) rm -f /tmp/dirip.$$ ;  rm -f /tmp/finallist.$$ ;
         rm -f /tmp/file2delete.tmp.$$; return;;
esac
 rm -f /tmp/dirip.$$
 rm -f /tmp/finallist.$$
 rm -f /tmp/file2delete.tmp.$$
 return
}



while true
do
dialog --clear --title "Main Menu" \
        --menu "To move [UP/DOWN] arrow keys \n\
[Enter] to Select\n\
        Choose the Service you like:" 20 51 4 \
        "Date/time"       "To see System Date & Time" \
        "Calender"        "To see Calaender"\
                "Delete"        "To remove file"\
                "Exit"          "To exit this Program" 2> menuchoice.temp.$$

retopt=$?

choice=`cat menuchoice.temp.$$`

rm -f menuchoice.temp.$$

case $retopt in
    0)
                case $choice in
                    Date/time) show_datetime ;;
                    Calender) show_cal ;;
                    Delete) delete_file ;;
                    Exit) exit 0;;
```

```
        esac
       ;;
      1) exit ;;
      255) exit ;;
 esac
done
clear

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

20.

```
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
# Q19
#

nouser=`who | wc -l`
echo -e "User name: $USER (Login name: $LOGNAME)" >> /tmp/info.tmp.01.$$$
echo -e "Current Shell: $SHELL"  >> /tmp/info.tmp.01.$$$
echo -e "Home Directory: $HOME" >> /tmp/info.tmp.01.$$$
echo -e "Your O/s Type: $OSTYPE" >> /tmp/info.tmp.01.$$$
echo -e "PATH: $PATH" >> /tmp/info.tmp.01.$$$
echo -e "Current directory: `pwd`" >> /tmp/info.tmp.01.$$$
echo -e "Currently Logged: $nouser user(s)" >> /tmp/info.tmp.01.$$$

if [ -f /etc/redhat-release ]
then
```

```
   echo -e "OS: `cat /etc/redhat-release`" >> /tmp/info.tmp.01.$$$
fi

if [ -f /etc/shells ]
then
   echo -e "Available Shells: " >> /tmp/info.tmp.01.$$$
   echo -e "`cat /etc/shells`"  >> /tmp/info.tmp.01.$$$
fi

if [ -f /etc/sysconfig/mouse ]
then
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "Computer Mouse Information: " >> /tmp/info.tmp.01.$$$
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "`cat /etc/sysconfig/mouse`" >> /tmp/info.tmp.01.$$$
fi
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer CPU Information:" >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/cpuinfo >> /tmp/info.tmp.01.$$$

echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer Memory Information:" >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/meminfo >> /tmp/info.tmp.01.$$$

if [ -d /proc/ide/hda ]
then
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "Hard disk information:" >> /tmp/info.tmp.01.$$$
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "Model: `cat /proc/ide/hda/model` " >> /tmp/info.tmp.01.$$$
   echo -e "Driver: `cat /proc/ide/hda/driver` " >> /tmp/info.tmp.01.$$$
   echo -e "Cache size: `cat /proc/ide/hda/cache` " >> /tmp/info.tmp.01.$$$
fi
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
```

```
echo -e "File System (Mount):" >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/mounts >> /tmp/info.tmp.01.$$$

if which dialog > /dev/null
then
   dialog  --backtitle "Linux Software Diagnostics (LSD) Shell Script Ver.1.0" --title "Press
Up/Down Keys to move" --textbox  /tmp/info.tmp.01.$$$ 21 70
else
   cat /tmp/info.tmp.01.$$$ |more
fi

rm -f /tmp/info.tmp.01.$$$

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

21.

```
For2
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

echo "Can you see the following:"

for (( i=1; i<=5; i++ ))
do
   for (( j=1; j<=i;  j++ ))
   do
```

```
  echo -n "$i"
 done
 echo ""
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#

For3
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

echo "Can you see the following:"

for (( i=1; i<=5; i++ ))
do
 for (( j=1; j<=i;  j++ ))
 do
  echo -n "$j"
 done
 echo ""
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

```
For4
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

echo "Climb the steps of success"

for (( i=1; i<=5; i++ ))
do
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " |"
   done
   echo "_ "
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#

For5
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
```

```
echo "Stars"

for (( i=1; i<=5; i++ ))
do
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " *"
   done
   echo ""
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#

For6
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

echo "Stars"

for (( i=1; i<=5; i++ ))
do
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " *"
   done
   echo ""
done
```

```
for (( i=5; i>=1; i-- ))
do
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " *"
   done
   echo ""
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#



For6
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

clear

for (( i=1; i<=3; i++ ))
do
   for (( j=1; j<=i;  j++ ))
   do
    echo -n "|Linux"
   done
   echo "______"
done
```

```
for (( i=3; i>=1; i-- ))
do
   for (( j=1; j<=i;  j++ ))
   do
    echo -n "|Linux"
   done

   if [ $i -eq 3 ]; then
     echo -n "______"
     echo -n -e ">> Powerd Server.\n"
   else
     echo "~~~~~"
   fi
done

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#



For7
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

MAX_NO=0

echo -n "Enter Number between (5 to 9) : "
read MAX_NO
```

```
if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
  echo "I ask to enter number between 5 and 9, Okay"
  exit 1
fi

clear

for (( i=1; i<=MAX_NO; i++ ))
do
   for (( s=MAX_NO; s>=i; s-- ))
   do
      echo -n " "
   done
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " $i"
   done
   echo ""
done

for (( i=1; i<=MAX_NO; i++ ))
do
   for (( s=MAX_NO; s>=i; s-- ))
   do
      echo -n " "
   done
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " ."
   done
   echo ""
done

echo -e "\n\n\t\t\tI hope you like it my stupidity (?)"
```

```
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#



For8
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#

MAX_NO=0

echo -n "Enter Number between (5 to 9) : "
read MAX_NO

if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
  echo "I ask to enter number between 5 and 9, Okay"
  exit 1
fi

clear

for (( i=1; i<=MAX_NO; i++ ))
do
   for (( s=MAX_NO; s>=i; s-- ))
   do
     echo -n " "
   done
   for (( j=1; j<=i;  j++ ))
   do
```

```
    echo -n " $i"
   done
   echo ""
done

for (( i=1; i<=MAX_NO; i++ ))
do
   for (( s=MAX_NO; s>=i; s-- ))
   do
     echo -n " "
   done
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " ."
   done
   echo ""
done

echo -e "\n\n\t\t\tI hope you like it my stupidity (?)"

#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#



For9
#!/bin/bash
#
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
#
# Written by Vivek G. Gite <vivek@nixcraft.com>
#
# Latest version can be found at http://www.nixcraft.com/
#
```

```
MAX_NO=0

echo -n "Enter Number between (5 to 9) : "
read MAX_NO

if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
  echo "I ask to enter number between 5 and 9, Okay"
  exit 1
fi

clear

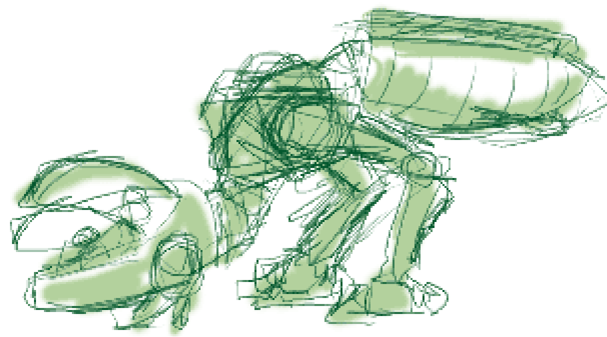
for (( i=1; i<=MAX_NO; i++ ))
do
   for (( s=MAX_NO; s>=i; s-- ))
   do
     echo -n " "
   done
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " ."
   done
   echo ""
done
###### Second stage ######################
##
##
for (( i=MAX_NO; i>=1; i-- ))
do
   for (( s=i; s<=MAX_NO; s++ ))
   do
     echo -n " "
   done
   for (( j=1; j<=i;  j++ ))
   do
    echo -n " ."
```

```
    done
    echo ""
done



echo -e "\n\n\t\t\tI hope you like it my stupidity (?)"


#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```





ภาคผนวก ก

การติดตั้งระบบปฏิบัติการ FreeBSD

คำแนะนำก่อนการติดตั้ง

ในการติดตั้งระบบปฏิบัติการ FreeBSD นั้น จะต้องจัดเตรียมเครื่องมือต่างๆ ที่จำเป็นสำหรับการติดตั้งไว้ก่อน ดังต่อไปนี้

- เครื่องคอมพิวเตอร์ตระกูล x86 ความเร็วของซีพียู ขนาดของหน่วยความจำ ขนาดของฮาร์ดดิสก์ ชนิดของการ์ดเน็ตเวิร์ค และจำนวนจำนวนของการ์ดที่ใช้งาน สามารถอ่านเพิ่มเติมได้ในบทที่ 2 (ความต้องการต่ำสุดคือ ความเร็วของซีพียูไม่ควรต่ำกว่า 1 GHz, หน่วยความจำไม่ต่ำกว่า 512 MB, ขนาดของฮาร์ดดิสก์ไม่น้อยกว่า 1 GB, และจำนวนการ์ดเน็ตเวิร์คขึ้นอยู่กับจำนวนของเน็ตเวิร์คที่ต้องการ)
- BIOS สามารถกำหนดให้ boot จาก CD-ROM ได้ (ถ้าสามารถ boot จาก USB, Compact Flash ได้ด้วยจะดีมาก)
- แผ่นระบบปฏิบัติการสำหรับติดตั้ง (FreeBSD)
- จำเป็นต้องเชื่อมต่ออินเทอร์เน็ต เพื่อติดตั้งซอฟต์แวร์เพิ่มเติม เช่น gcc, gmake เป็นต้น
- ดาวน์โหลด FreeBSD จาก http://www.freebsd.org/where.html เป็นไฟล์ ISO ที่เป็นแผ่นชนิด CD จำนวนประมาณ 3 แผ่น (7.0-RELEASE-i386-disc1.iso, 7.0-RELEASE-i386-disc2.iso, 7.0-RELEASE-i386-disc3.iso)
- แตกไฟล์ที่บีบอัดมาออกด้วยโปรแกรมประเภท winzip, winrar (ถ้ามีการบีบอัดข้อมูลไว้)
- เขียนไฟล์ Image ที่ดาวน์โหลดมาลงแผ่น CD (ใช้โปรแกรม burn image เช่น Nero, PowerISO, Alcohol 120 เป็นต้น)
- ใส่แผ่นติดตั้งใน CD
- รีเซ็ตเครื่องคอมพิวเตอร์ ในระหว่างคอมพวเตอร์เริ่มทำงานให้กดปุ่ม F2 เพื่อเข้าไปเซ็ตไบออส ในเมนูลำดับการ boot ให้เลือก Boot CD-ROM เป็นอันดับแรก restart เครื่องอีกครั้ง

ขั้นตอนการติดตั้ง FreeBSD

ในหนังสือเล่มนี้จะใช้ FreeBSD เวอร์ชัน 7.0 (ใส่แผ่น CD แผ่นที่ 1 ไปยัง CD-ROM) เนื่องจากเวลาที่เขียนหนังสือเล่มนี้ เวอร์ชันดังกล่าวเสถียรที่สุด ในอนาคตอาจจะมีเวอร์ชันใหม่เผยแพร่ออกมาก็สามารถใช้งานแทนเวอร์ชันดังกล่าวได้เช่นเดียวกัน และมีขั้นตอนการติดตั้งคล้ายๆ กัน การติดตั้งมีขั้นตอนดังนี้

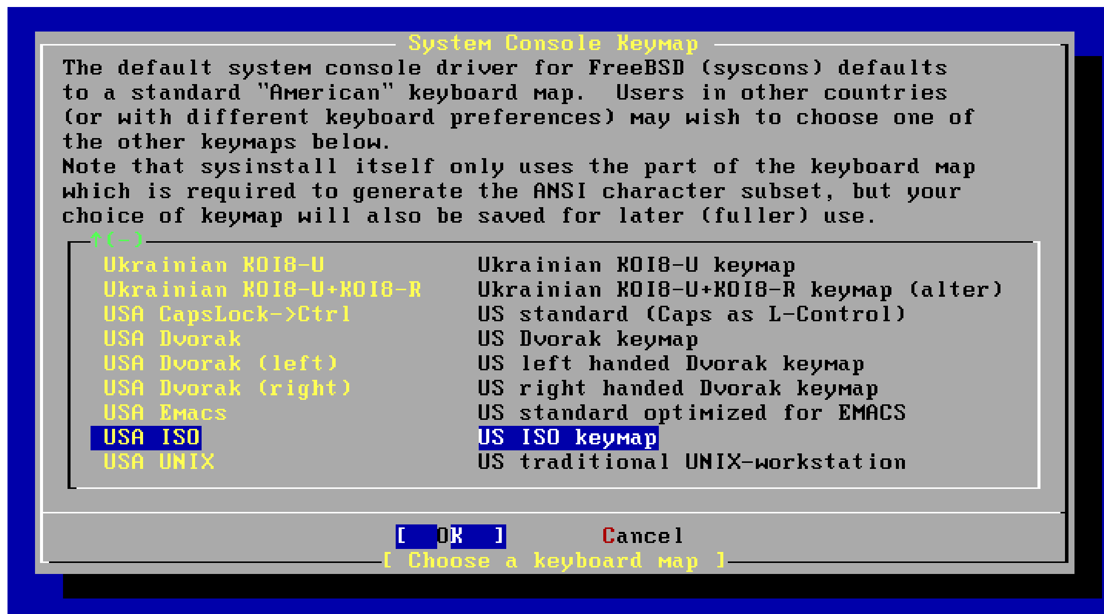
- เลือกวิธีการ Boot ระบบ (เลือกหมายเลข 1 หรือกดปุ่ม Enter)



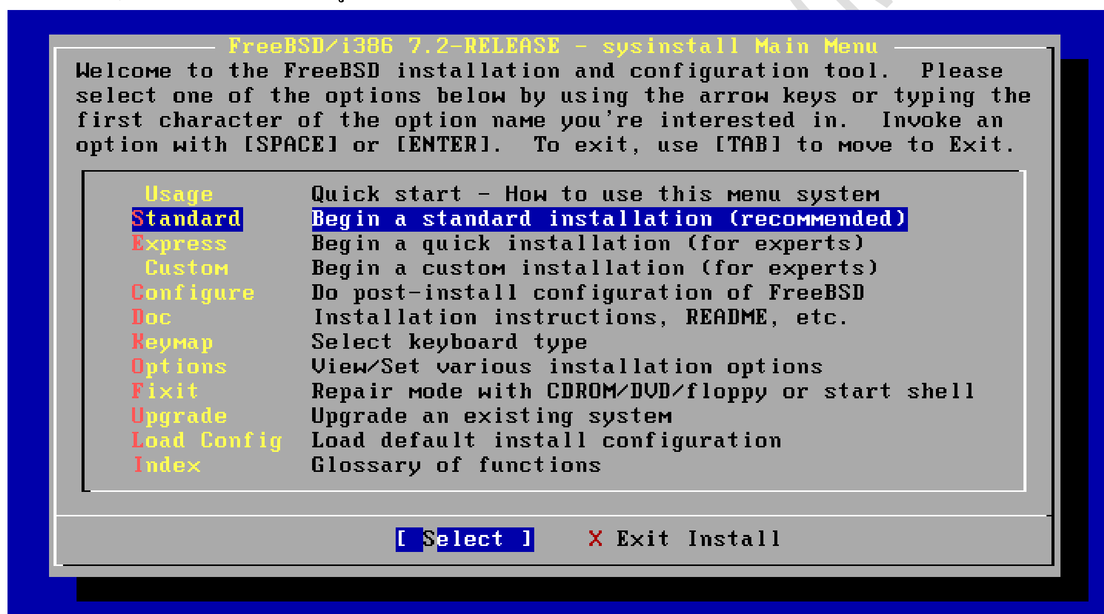
- Country Selection: เลือกประเทศไทย 218 Thailand



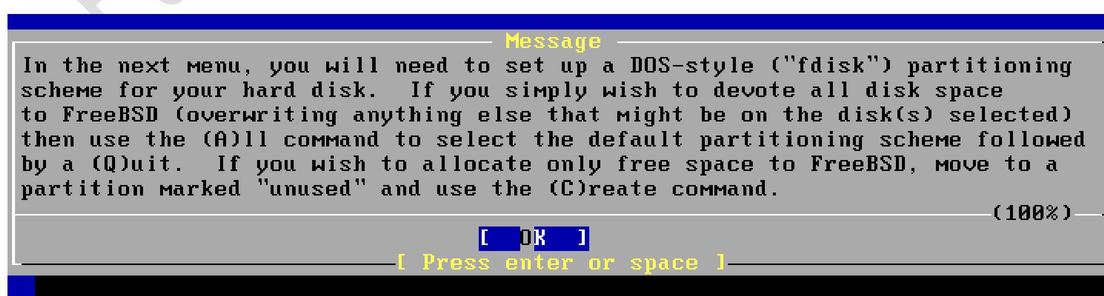
- System Console Keymap: เลือกรูปแบบคีย์บอร์ดที่ใช้งาน เลือก USA ISO



- Sysinstall Main Menu: รูปแบบการติดตั้ง เลือกติดตั้งแบบ Stardard



- Message: แสดงให้ทราบว่าขั้นตอนต่อไปจะต้องทำการ fdisk เพื่อจัดการ partition ให้เหมาะสมสำหรับติดตั้ง FreeBSD เลือก OK



- Fdisk Partition Editor: จัดการพื้นที่ฮาร์ดดิสก์ด้วยโปรแกรม fdisk เลือก A คือใช้ฮาร์ดดิสก์ทั้งหมดในการติดตั้ง FreeBSD

```
Disk name:      ad0                                   FDISK Partition Editor
DISK Geometry:  6241 cyls/16 heads/63 sectors = 6290928 sectors (3071MB)

Offset       Size(ST)        End     Name  PType       Desc  Subtype    Flags

        0    6291456    6291455        -     12     unused        0



The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice    F = 'DD' mode
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable    | = Wizard m.
T = Change Type        U = Undo All Changes     Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

```
Disk name:      ad0                                   FDISK Partition Editor
DISK Geometry:  6241 cyls/16 heads/63 sectors = 6290928 sectors (3071MB)

Offset       Size(ST)        End     Name  PType       Desc  Subtype    Flags

        0          63         62        -     12     unused        0
       63     6290865    6290927    ad0s1      8    freebsd      165
  6290928         528    6291455        -     12     unused        0



The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice    F = 'DD' mode
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable    | = Wizard m.
T = Change Type        U = Undo All Changes     Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

- Install Boot Manager..: ติดตั้งรูปแบบเริ่มต้นของการ Boot เลือก Standard

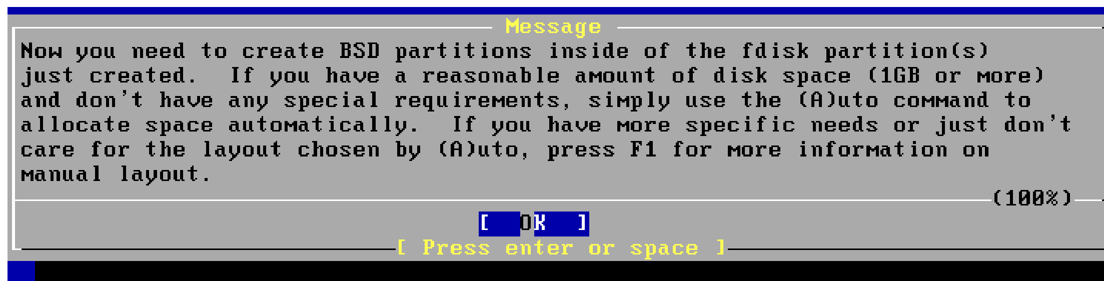
```
                    Install Boot Manager for drive ad0?
 FreeBSD comes with a boot selector that allows you to easily
 select between FreeBSD and any other operating systems on your machine
 at boot time.  If you have more than one drive and want to boot
 from the second one, the boot selector will also make it possible
 to do so (limitations in the PC BIOS usually prevent this otherwise).
 If you do not want a boot selector, or wish to replace an existing
 one, select "standard".  If you would prefer your Master Boot
 Record to remain untouched then select "None".

   NOTE:  PC-DOS users will almost certainly require "None"!

          BootMgr    Install the FreeBSD Boot Manager
          Standard   Install a standard MBR (no boot manager)
          None       Leave the Master Boot Record untouched

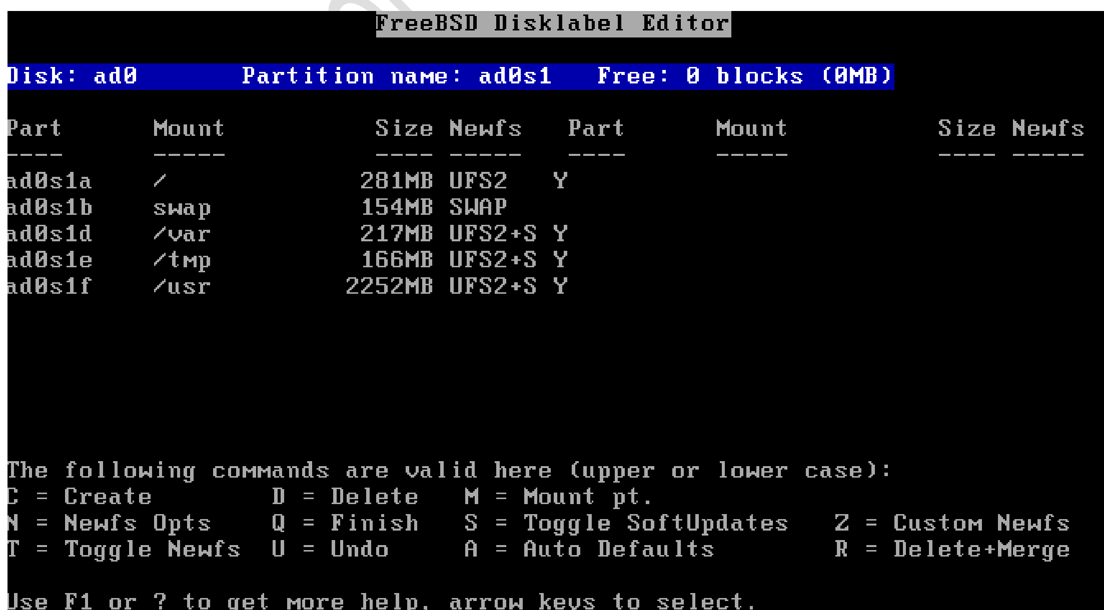
                      [  OK  ]        Cancel
                 [ Press F1 to read about drive setup ]
```

- Message: ปรับแต่ง Partition โดยละเอียด เลือก OK

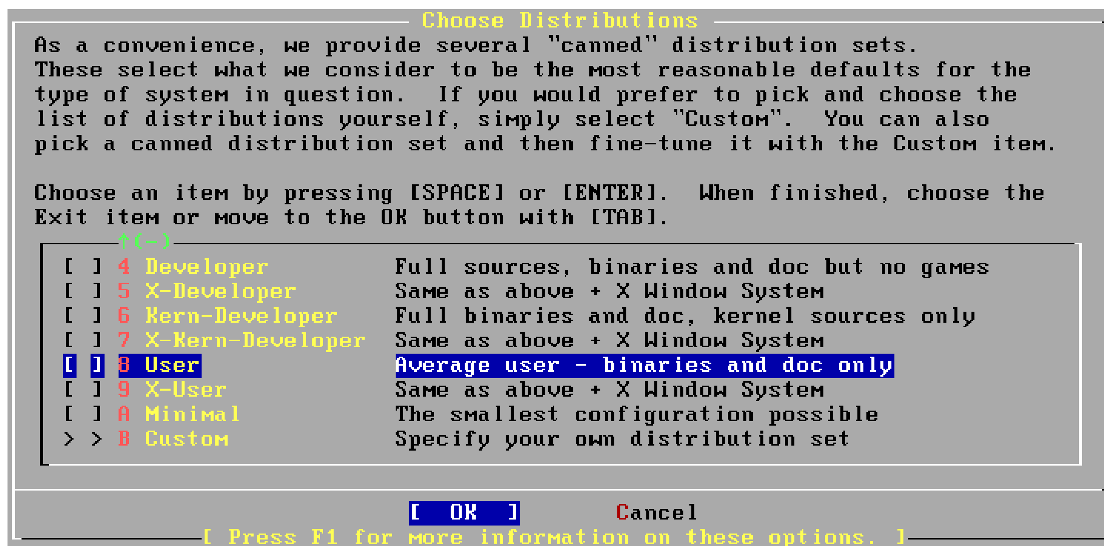


- FreeBSD Disklabel Editor: สร้าง partition ย่อยที่ FreeBSD จำเป็นต้องใช้ เลือก A ระบบจะเป็นผู้สร้างให้ และคำนวณพื้นที่ที่จำเป็นแบบอัตโนมัติ ถ้าผู้ใช้ต้องการสร้างเองให้เลือก C คือ Create แล้วจึงกำหนดเนื้อที่ตามที่ต้องการเอง จากนั้นกด Q

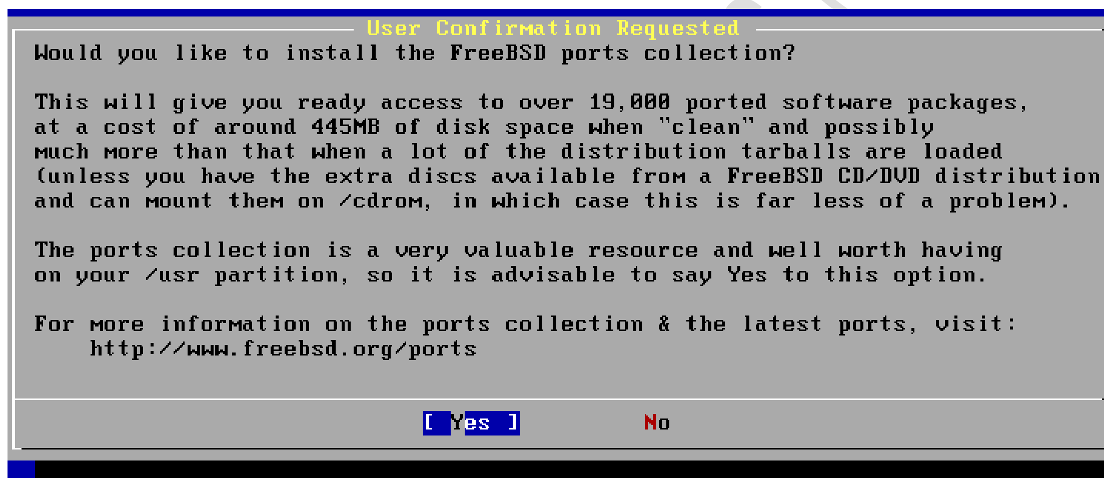




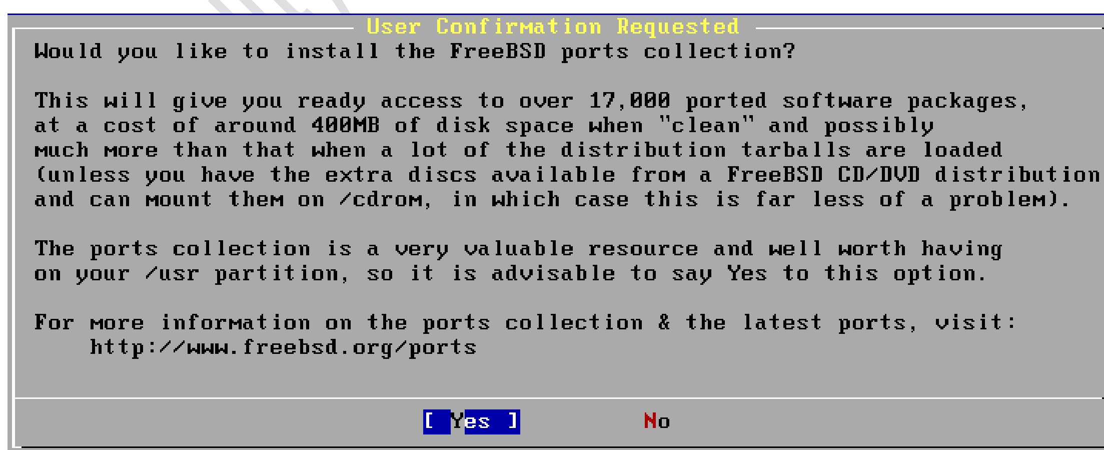
- Choose Distribution: แสดง Package การติดตั้งตามความเหมาะสมกับงานที่ต้องการ เลือก 8 User เพราะจำเป็นต้องมีการคอมไพล์โปรแกรมเมื่อติดตั้งเราท์เตอร์



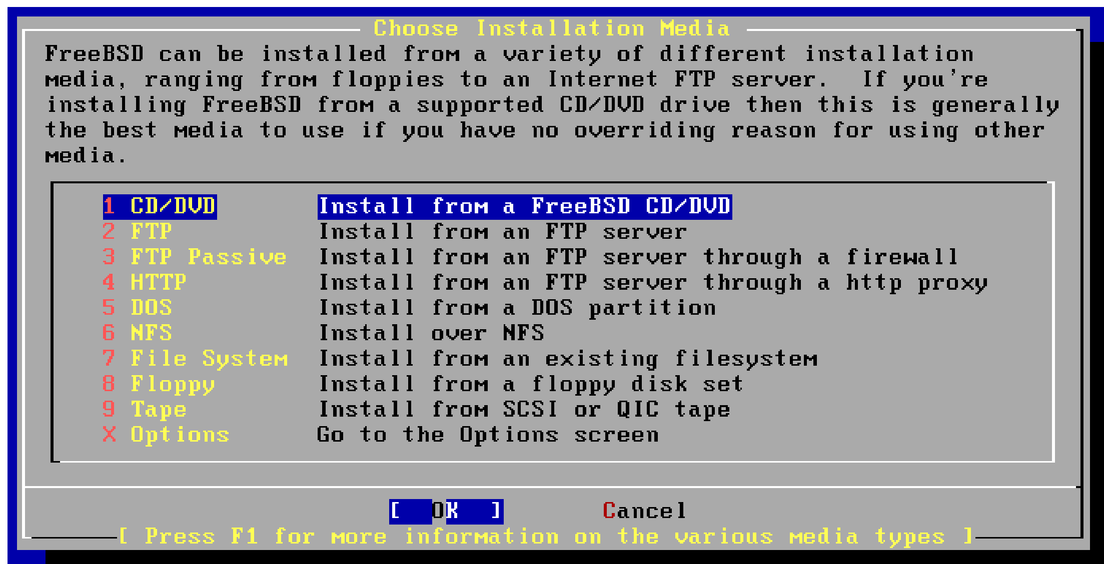
- User Comfirmation Requested: ยืนยันการติดตั้งอีกครั้ง เมื่อเลือกแบบ User จำเป็นต้องใช้เนื้อที่ในโฟร์เดอร์ /usr/ports ซึ่งเก็บซอฟต์แวร์สำหรับติดตั้งประมาณ 400 MB สามารถอ่านรายละเอียดซอฟต์แวร์ทั้งหมด ได้จาก www.freebsd.org/ports จากนั้นเลือก Yes



- ขั้นตอนต่อไปเลือก <<< X Exit



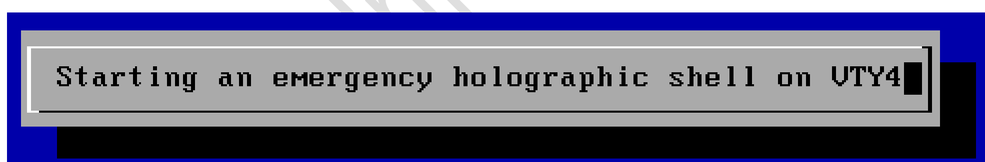
- Choose Installation Media: รูปแบบสื่อที่ใช้ติดตั้ง เลือกติดตั้งด้วย CD/DVD



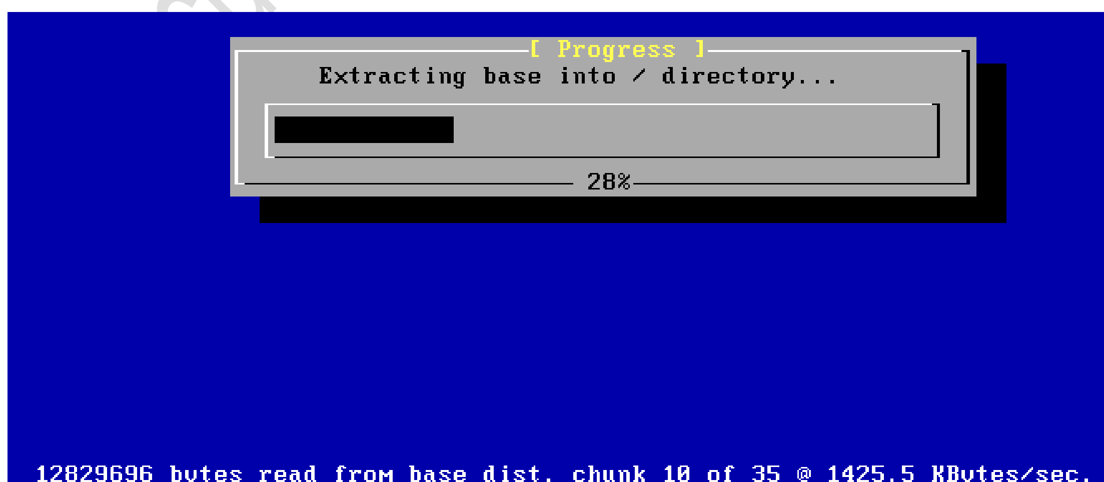
- User Comfirmation Requested: ยืนยันการติดตั้งอีกครั้ง เลือก Yes



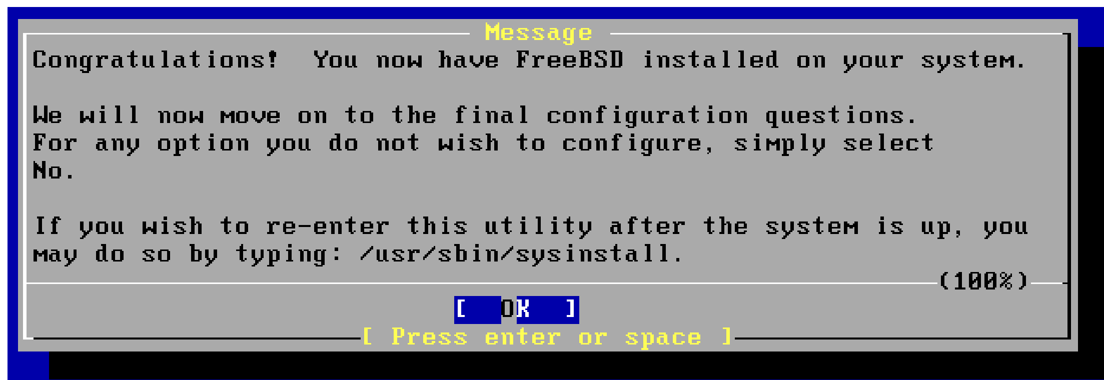
- Installer จะทำการ format และเริ่มทำการติดตั้ง



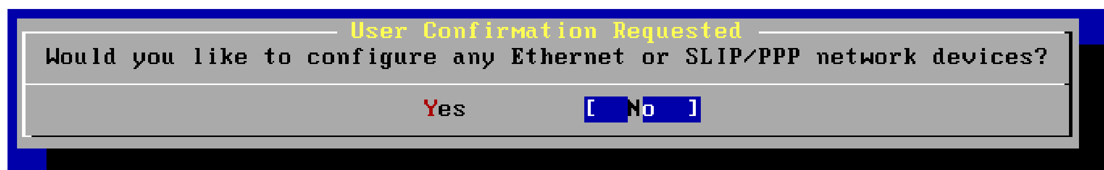
- เริ่มทำการคัดลอกระบบปฏิบัติการลงบนฮาร์ดดิสก์ ใช้เวลาประมาณ 15-30 นาที ขึ้นอยู่กับความเร็วของเครื่องและจำนวนซอฟต์แวร์ที่เลือกติดตั้ง



- เมื่อคัดลอกระบบปฏิบัติการเสร็จ Installer จะแจ้งว่า สามารถปรับแต่งระบบเพิ่มเติมหลังจากการติดตั้งได้ โดยการเรียกโปรแกรม sysinstll ซึ่งติดตั้งอยู่ใน /usr/sbin/sysinstall



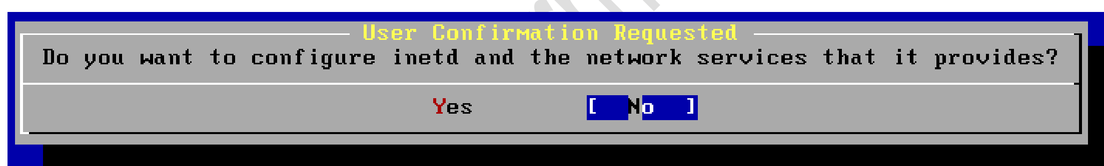
- User Comfirmation Requested: ต้องการคอนฟิก SLIP/PPP บนเครื่องหรือไม่ เลือก No



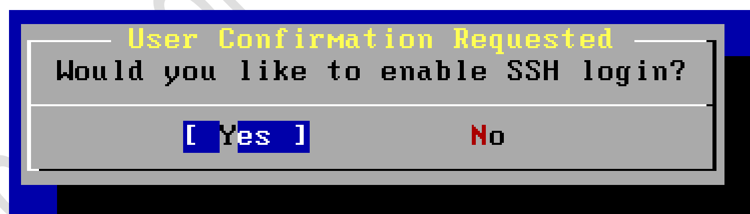
- ต้องการติดตั้งเครื่องให้เป็น Network Gateway หรือไม่ เลือก No



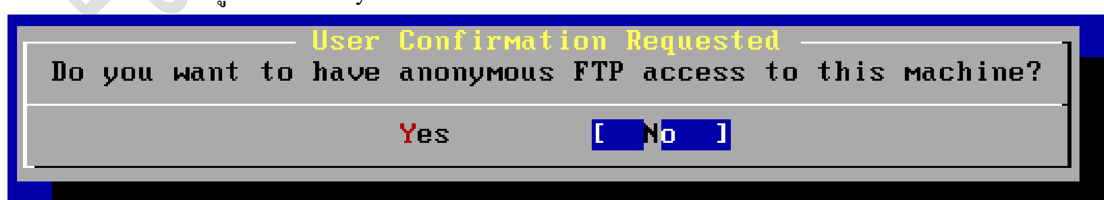
- ต้องการคอนฟิก inetd และ Network Services หรือไม่ เลือก No



- ต้องการเปิดการใช้งาน remote secure shell (SSH) หรือไม่ เลือก Yes



- ต้องการให้ผู้ใช้แบบ anonymous สามารถเข้าใช้งานเครื่องได้หรือไม่ เลือก No



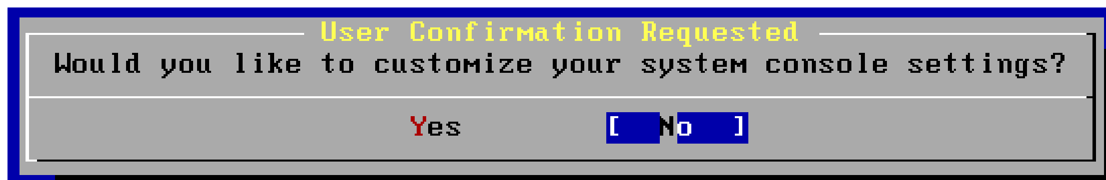
- ต้องการเปิดให้ทำงานเป็น NFS server หรือไม่ เลือก No



- ต้องการเปิดให้ทำงานเป็น NSF Client หรือไม่ เลือก No



- ต้องการปรับแต่ง console ที่ใช้งานหรือไม่ เลือก No



- ต้องการตั้งเวลาเครื่องหรือไม่ เลือก Yes



- เลือกเวลาอ้างอิงในการใช้งาน ถ้าเลือก local จะใช้เวลาในของประเทศไทย ถ้าใช้ UTC จะเทียบกับเวลามาตรฐานโลก เลือก No



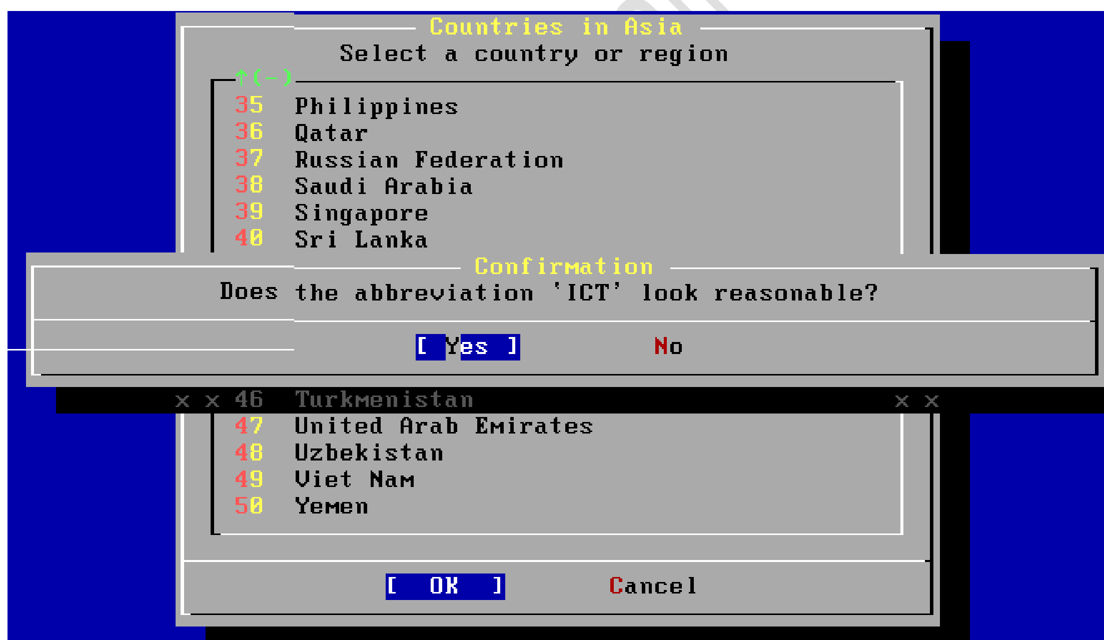
- Time Zone Selector: เลือกใช้เวลาของแต่ละทวีป เลือก 5 Asia



- Countries in Asia: เลือกประเทศที่อยู่ในทวีปเอเชีย เลือก 44 Thailand



- Confirmation เลือก Yes



- ติดตั้งซอฟต์แวร์ที่ใช้พัฒนาร่วมกับลินุกซ์หรือไม่ เลือก Yes



```
Adding packages/All/linux_base-fc-4_14.tbz
from acd0
```

ซอฟต์แวร์ Linux binary compatibility ทำการติดตั้งลงบนฮาร์ดดิสก์

- ต้องการติดตั้งอินเทอร์เฟสแบบ PS2, Serial หรือ Mouse เพิ่มเติมหรือไม่ เลือก No

```
User Confirmation Requested
Does this system have a PS/2, serial, or bus mouse?

Yes            [ No ]
```

- แสดงข้อความแจ้งว่ายังมีซอฟต์แวร์อีกมากมายให้เลือกใช้งานได้ ต้องการดูข้อมูลเพิ่มเติมหรือไม่ เลือก Yes

```
User Confirmation Requested
The FreeBSD package collection is a collection of thousands of ready-to-run
applications, from text editors to games to WEB servers and more.  Would you
like to browse the collection now?

[ Yes ]          No
```

- แจ้งเตือนว่าควรจะมีการสร้างผู้ใช้งานอื่นๆ เพิ่มเติม ไม่ควรใช้ user root ในการทำงานประจำ เพื่อความปลอดภัยของระบบ เลือก Yes

```
User Confirmation Requested
Would you like to add any initial user accounts to the system?
Adding at least one account for yourself at this stage is suggested
since working as the "root" user is dangerous (it is easy to do
things which adversely affect the entire system).

[ Yes ]          No
```

- ถ้าต้องการสร้าง user จะต้องทำการสร้าง group ก่อน จากนั้นจึงสร้าง user เมื่อสร้างเสร็จแล้วให้เลือก Exit

```
User and group management
The submenus here allow to manipulate user groups and
login accounts.

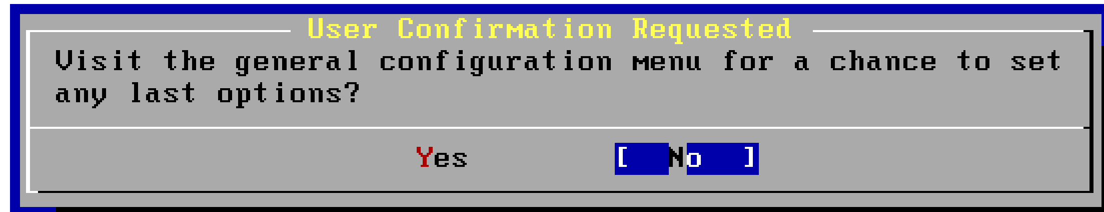
X Exit    Exit this menu (returning to previous)
User      Add a new user to the system.
Group     Add a new user group to the system.

[ OK ]          Cancel
[ Configure your user groups and users ]
```

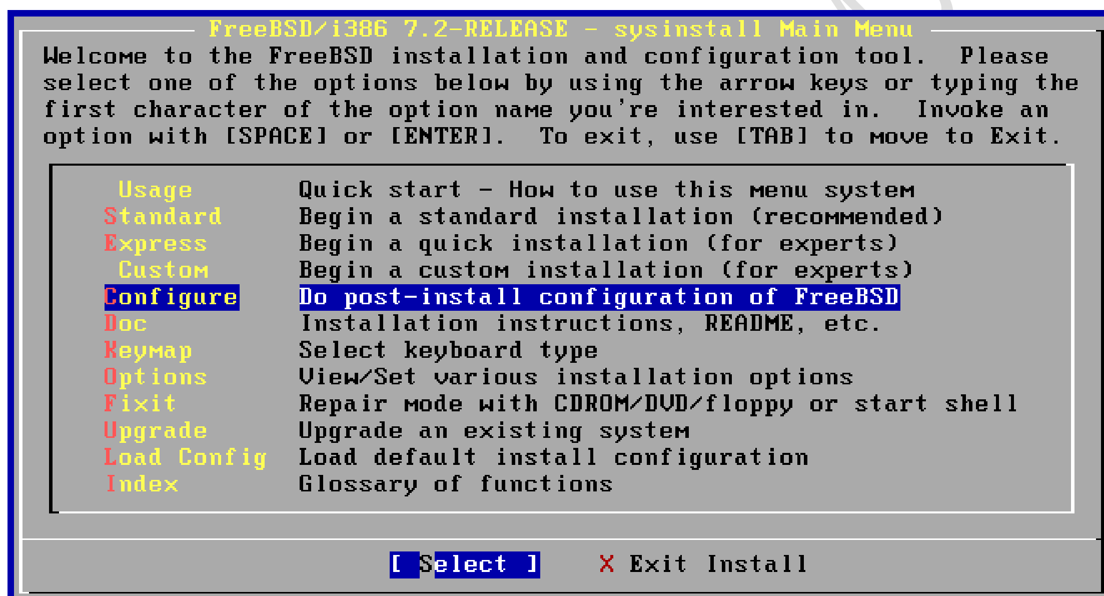
- ตั้งรหัสผ่านให้กับ root เลือก OK

```
Message
Now you must set the system manager's password.
This is the password you'll use to log in as "root".
                                            (100%)
[ OK ]
[ Press enter or space ]
```

- ใส่รหัสผ่านที่ต้องการ ความยาวของรหัสไม่ควรต่ำกว่า 6 ตัวอักษร โดยใช้ ตัวอักษร + ตัวเลข + สัญลักษณ์พิเศษ

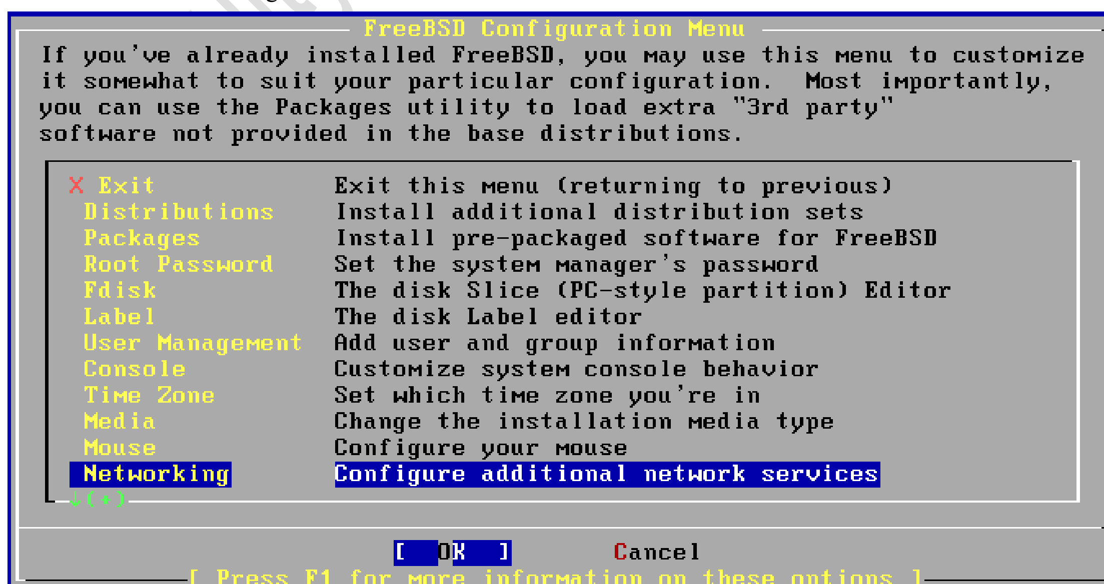


- ต้องการตรวจสอบข้อมูลการติดตั้งครั้งสุดท้ายหรือ เลือก No

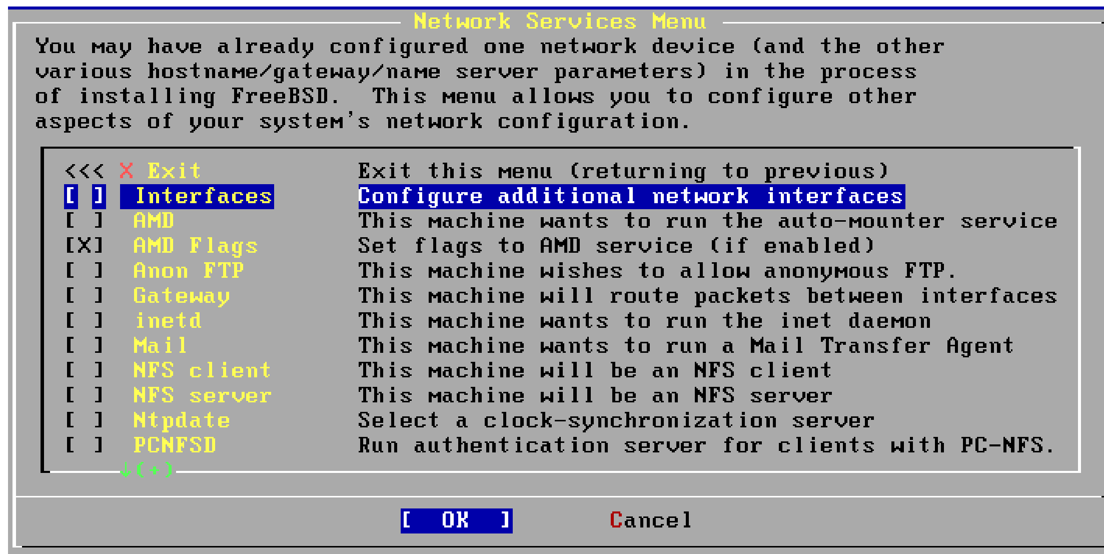


- เมื่อกระบวนการติดตั้งเสร็จ Installer จะกลับไปเมนู Sysinstall Main Menu อีกครั้งเพื่อให้ผู้ใช้สามารถปรับแต่งการติดตั้งได้ใหม่อีกครั้ง หรือแม้แต่ต้องการติดตั้งใหม่ทั้งหมดก็สามารถทำได้ เลือก Configure เพื่อทำการติดตั้งเน็ตเวิร์คต่อไป

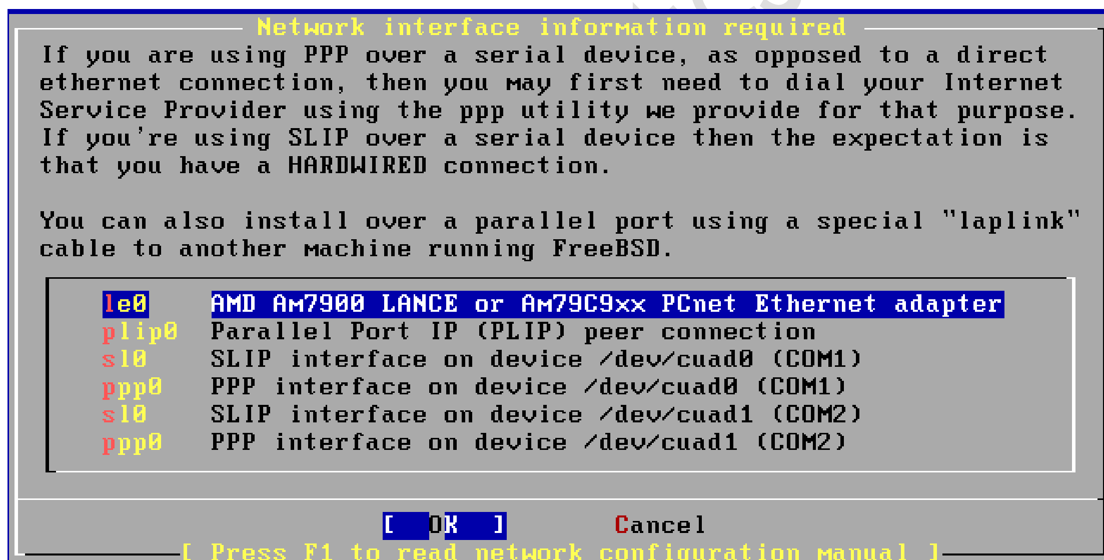
- เลือก Networking เพื่อปรับแต่งเครือข่าย เนื่องจากจำเป็นต้องมีการติดตั้งซอฟต์แวร์ผ่านอินเทอร์เน็ต



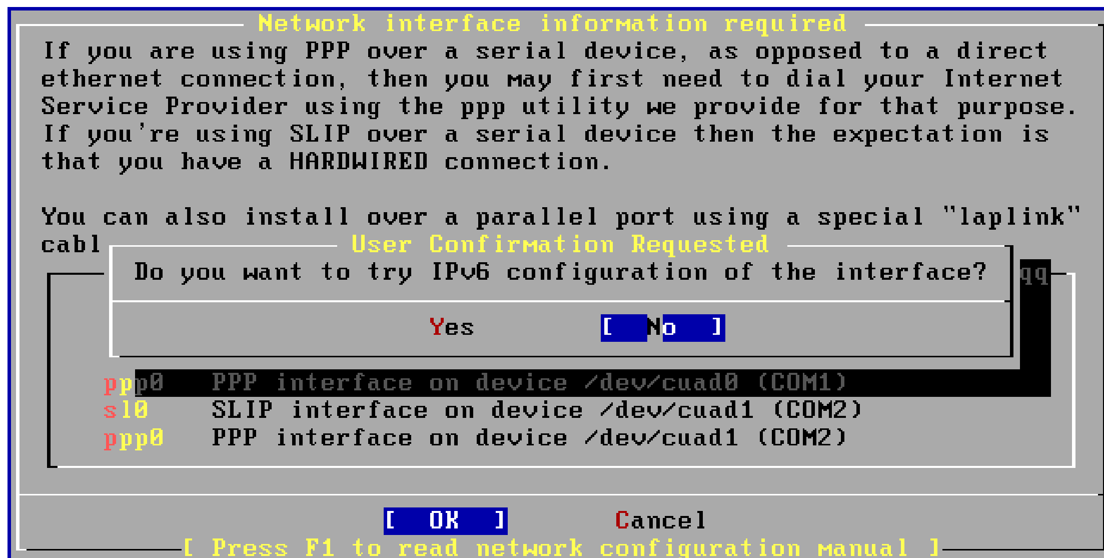
- เลือก [X] Interfaces โดยเลื่อนลูกศรไปยังเมนูดังกล่าวและกดปุ่ม spac bar (ถ้าต้องการข้ามไปเมนูอื่นๆ ให้กดปุ่ม Tab)



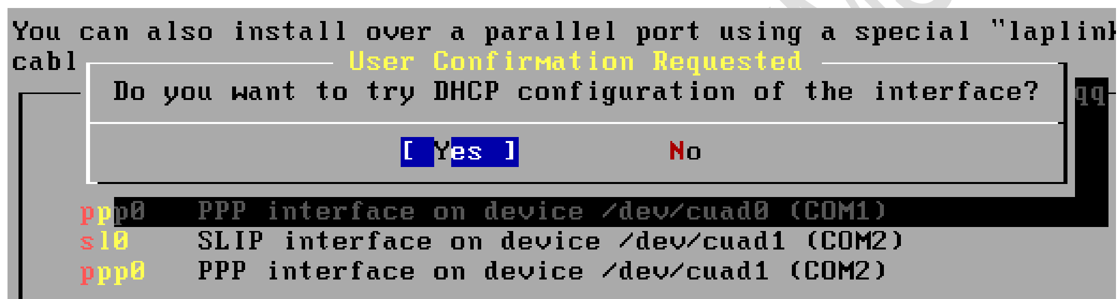
- เลือกอินเทอร์เฟสที่ต้องการคอนฟิก เลือก le0 (โดยปกติ freebsd จะมองเห็นอินเทอร์เฟสชนิดอีเทอร์เน็ต เป็น lex โดยที่ x หมายถึงลำดับของการ์ดเน็ตเวิร์ดที่ติดตั้งบนเครื่อง)



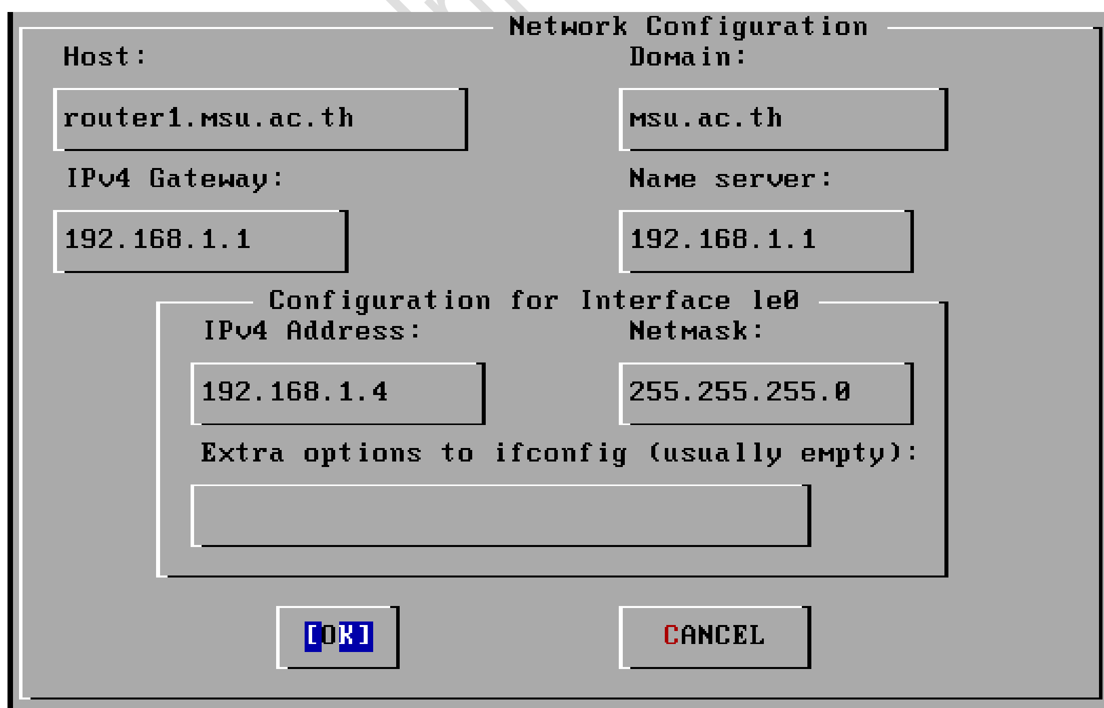
- ต้องการคอนฟิก IPv6 หรือไม่ เลือก No



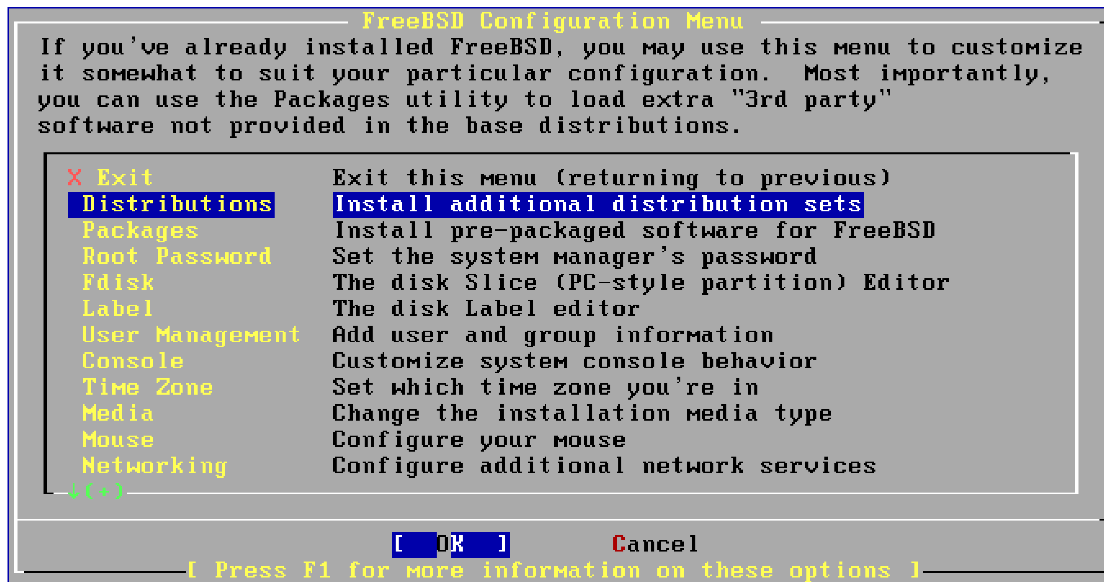
- ต้องการรับ IP Address จาก DHCP server หรือไม่ เลือก Yes (ถ้าต้องการกำหนด IP เองให้เลือก No)

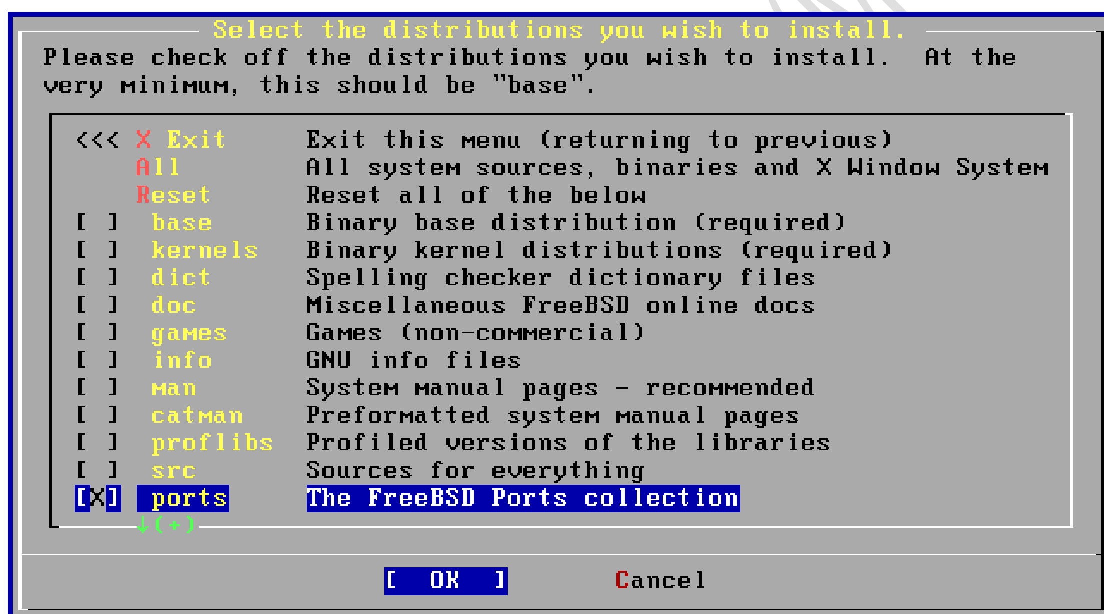


- กำหนด IP Address ให้กับการ์ดเน็ตเวิร์ค พร้อมชื่อเครื่อง โดเมนเนม เกตเวย์ subnet เมื่อกำหนดครบแล้วให้เลือก OK ต่อจากนั้นเลือก <<< X Exit



- เลือก Distributions จากนั้นเลือก ports แล้วกดปุ่ม OK รอสักครู่ Installer จะทำการคัดลอก Makefile ที่จะใช้ติดตั้งโปรแกรมในอนาคตลงในไฟร์เดอร์ /usr/ports/





- เลือก X Exit จากนั้นเลือก Package → CD/DVD → devel แล้วเลือกซอฟต์แวร์ดังต่อไปนี้ เพื่อใช้สำหรับการคอมไพล์โปรแกรม

  perl-5.8.8_1
  libtool-1.5.24
  autoconf-2.61_2
  M4-1.4.9
  gmake-3.81_2
  gettext-0.16.1_3

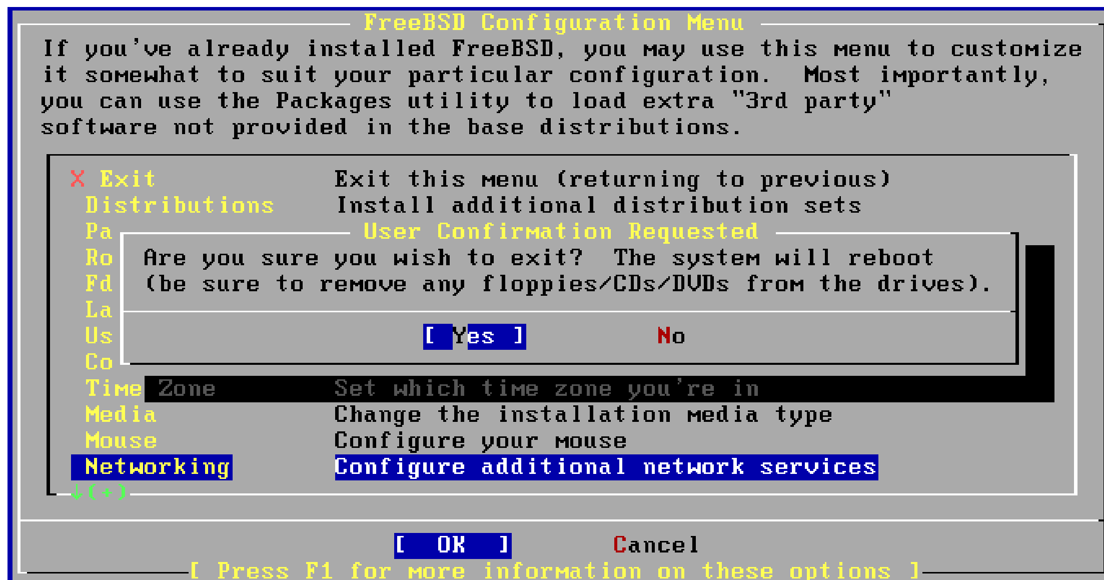
  และ lang

  Python25-2.5.1_1

  และ shells

  bash-3.2-25

- เลือก X Exit อีกครั้ง ขั้นตอนต่อไปให้เลือก [X Exit Install] Installer จะบอกให้ทราบว่าเครื่องจะมีการ restart ใหม่ ให้นำสื่อ CD ที่ติดตั้งออกจากเครื่องก่อน แล้วเลือก Yes



- เครื่องคอมพิวเตอร์จะเริ่มต้นทำงานใหม่อีกครั้ง รอสักครู่ จะปรากฏข้อความให้ทำการ login ให้ทำการ Login ด้วย user root ถึงขั้นตอนนี้ FreeBSD พร้อมที่จะใช้งาน

```
Sat Sep  5 19:58:11 ICT 2009

FreeBSD/i386 (router1.msu.ac.th) (ttyv0)

login: root
Password:
```

การติดตั้งระบบปฏิบัติการลินุกซ์ (Cent OS 5.2)

1. ตั้งค่า BIOS เครื่องคอมพิวเตอร์ หรือ server ให้ Boot เข้า CD-ROM เป็นอันดับแรก แล้วนำแผ่น CentOS 5.2 แผ่นที่ 1 ใส่ใน CD-ROM



เริ่มต้นการ Boot เข้าแผ่น CD-ROM

ในขั้นนี้จะพบหน้าจอการติดตั้ง CentOS5 ระบบจะให้เราเลือกรูปแบบการติดตั้ง ถ้าต้องการติดตั้งแบบ Graphic Mode ให้กด Enter

หรือถ้าเราต้องการติดตั้งแบบ Text Mode ให้พิมพ์คำว่า linux text แล้วกด Enter แต่ในที่นี้ผมแนะนำให้ติดตั้งแบบ Graphic Mode ทำได้โดยการ กด Enter ผ่านไปได้เลย

2. ระบบจะถามว่าคุณต้องการตรวจสอบแผ่นระบบปฏิบัติการก่อนการติดตั้งหรือไม่ซึ่งถ้าเราตรวจสอบก็จะใช้เวลานาน



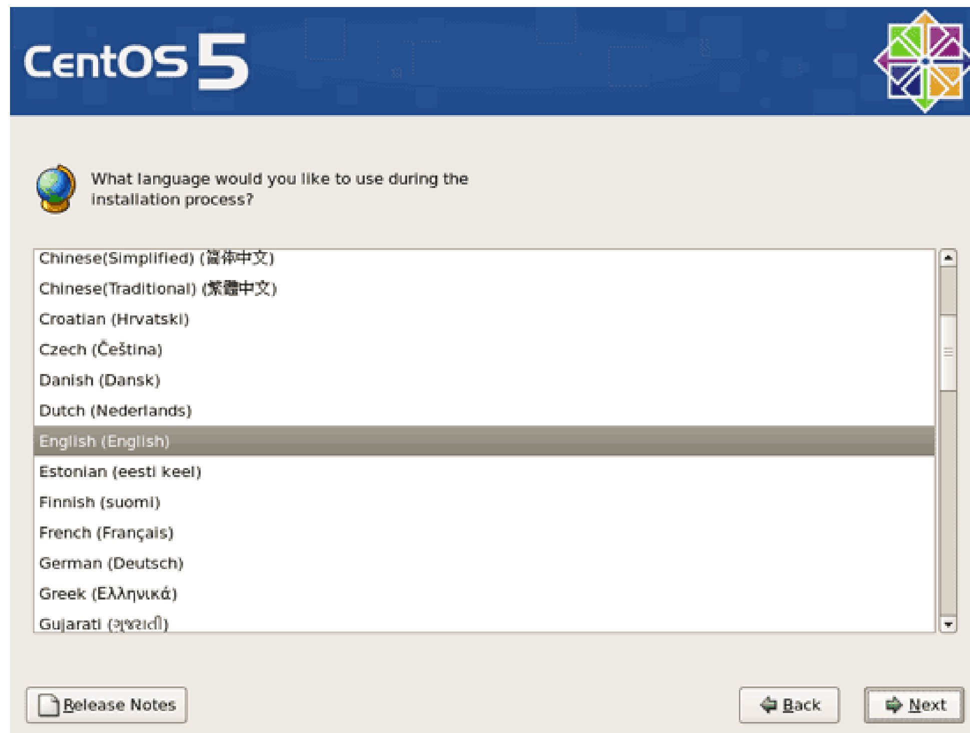
ในที่นี้ให้เลือก Skip แล้ว Enter เพื่อข้ามขั้นตอนการตรวจสอบแผ่น CD

3. เข้าสู่การติดตั้ง CentOS แบบGraphic Mode



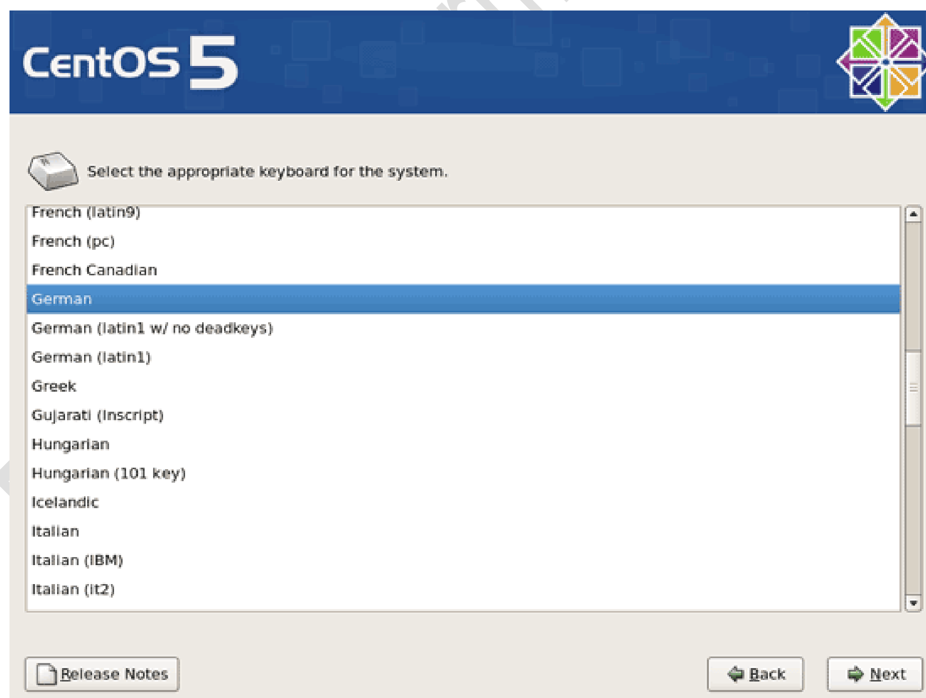
คลิกที่ปุ่ม Next เพื่อดำเนินการต่อไป

4. เลือกภาษามาตรฐานในการติดตั้ง



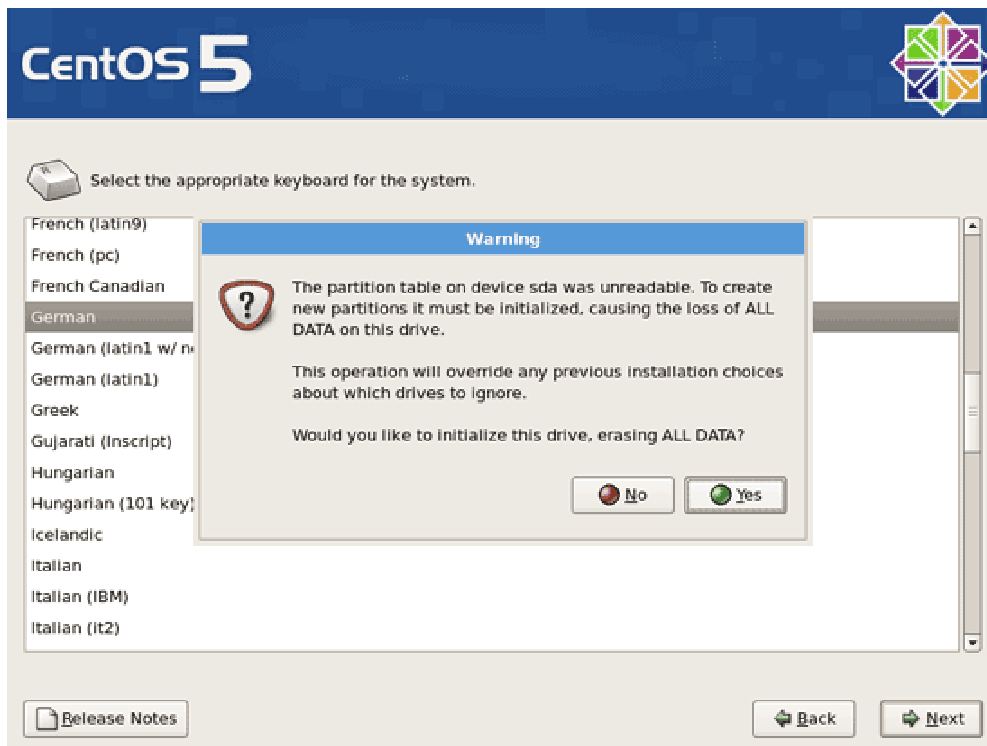
ให้เลือกภาษา English ดังภาพ แล้วคลิกที่ปุ่ม Next

5. เลือกภาษาให้คีย์บอร์ด



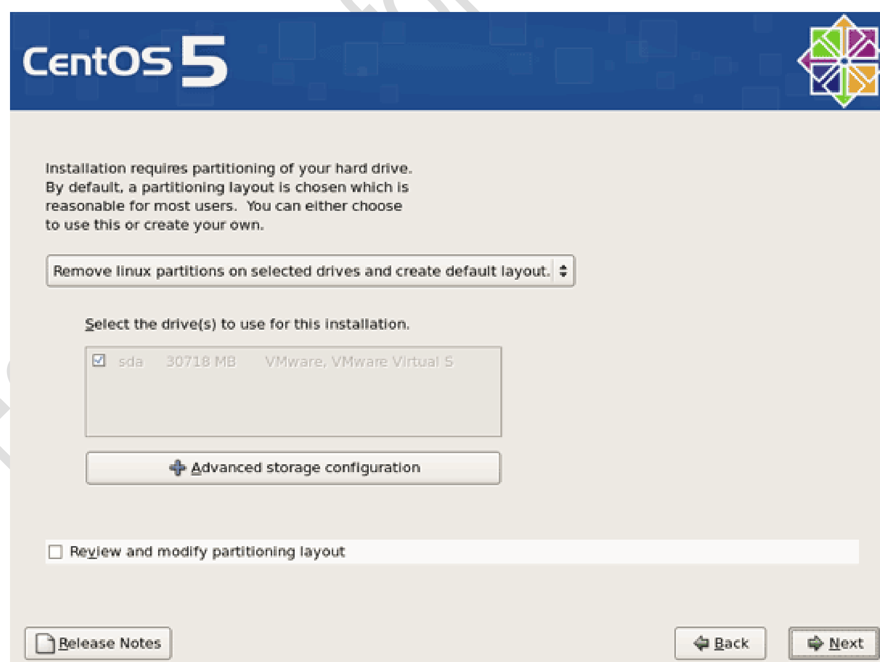
ให้เลือกภาษา English แล้วคลิกปุ่ม Next

6. ระบบจะแจ้งเตือนว่าการติดตั้งจะทำให้ข้อมูลที่มีอยู่เดิมลบหายไป



แจ้งเตือนการติดตั้งให้คลิกที่ปุ่ม Yes

7. เริ่มแบ่งพาติชั่น



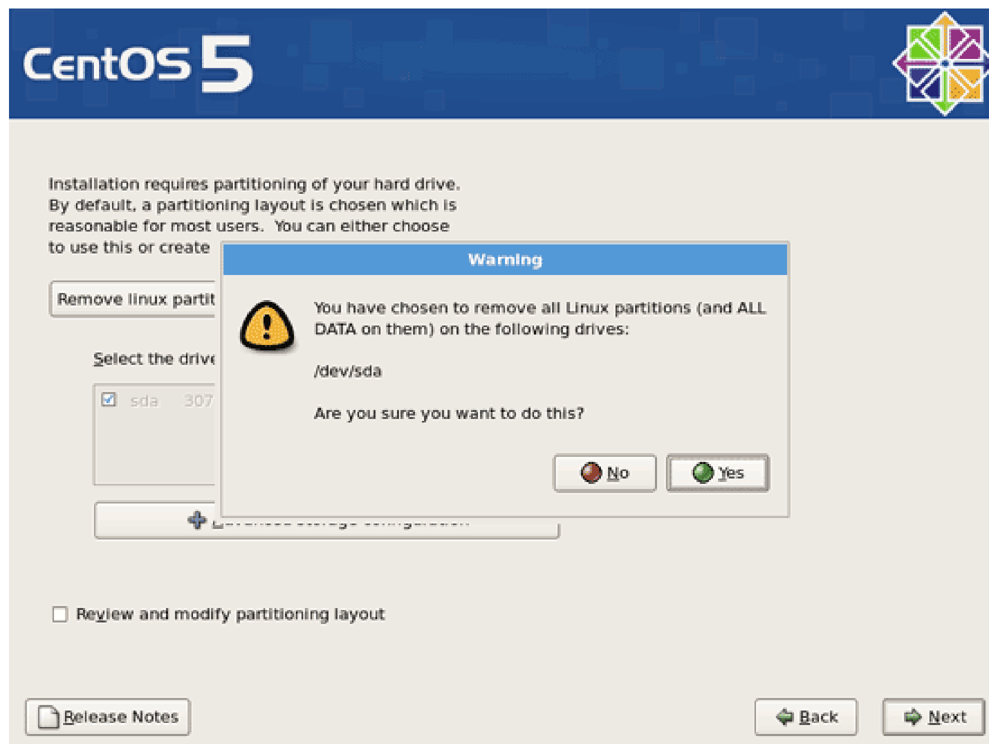
ในส่วนนี้เป็นการติดตั้งแบบกำหนดพาติชั่นเอง ให้เลือก Remove Custom ในช่องลิสรายการล่างสุดแล้วคลิกปุ่ม Next

การแบ่งพาติชั่นก็เหมือนกับการแบ่งไดว์การทำงานนั่นเองครับเพียงแต่ในลีนุกส์จะมีมากไปหน่อย สำหรับมือใหม่ก็แนะนำให้เลือกเป็น Auto ครับในขั้นตอนนี้ผมไม่มีภาพมาให้ดูเอาเป็นว่าจะเขียนอธิบายก็แล้วกันการแบ่งพาติชั่นให้ทำการเลือกที่ ฮาร์ดดิสก์แล้วคลิกที่ปุ่ม Delete เพื่อทำการลบ พาติชั่นเดิมออกไปก่อน

การสร้างให้คลิกที่ปุ่ม NEW แล้วเลือกรูปแบบพาติชั่นเช่น /boot กำหนดขนาดพื้นที่เป็น 512 การกำหนดพื้นที่สามารถกำหนดได้ดังนี้

ตารางการแบ่งพาติชั่น CentOS5.2 Linux Server	
Patition	**จำนวนพื้นที่ที่แบ่ง**
/boot / Swap	256 MB (ตั้งค่าให้เป็น Force to be primary Patition) 512 MB (ตั้งค่าให้เป็น Force to be primary Patition) ถ้า RAM เครื่อง Server มีไม่ถึง 1 GB ให้แบ่งเป็น 512 MB ถ้ามากกว่าเป็น 1024 MB (ตั้งค่าให้เป็น Force to be primary Patition)
/tmp	512 MB ใช้เก็บไฟล์ชั่วคราว
/usr	5,300 MB ใช้เก็บไฟล์ package (ค่าสูงสุด)
/var	512 MB
/var/ftp	15,000 MB เอาไว้ทำ FTP server บริการไฟล์ข้อมูล (/ftp ให้พิมพ์เพิ่ม)
/var/lib	2,500 MB ใช้เก็บฐานข้อมูล SQL (/lib ให้พิมพ์เพิ่ม)
/var/spool/squid	3,072 MB ใช้เก็บ Proxy Server (/spool/squid ให้พิมพ์เพิ่ม)
/var/spool/mail	1204 MB ใช้เก็บเมล์ (/spool/mail ให้พิมพ์เพิ่ม)
/usr/local	1,500 MB
/opt	512 MB
/home	ให้เลือก Fill to maximum allowable siza (ที่เหลือทั้งหมด)

8. เมื่อกำหนดพื้นที่เสร็จแล้วให้คลิกที่ปุ่ม Next



ระบบจะแจ้งเตือนว่าต้องการให้ลีนุกส์ลบข้อมูลพาติชั่นเพื่อเตรียมการติดตั้งให้คลิกที่ปุ่ม Yes

9. การกำหนดหมายเลข IP Address ให้เครือข่าย



ล

ในภาพนี้มีการ์ดแลน 1 ใบ ถ้าท่านมี 2 ใบจะพบ eth1

CentOS 5

Edit Interface

Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
Hardware address: 00:0C:29:B1:97:E1

Enable IPv4 support
Dynamic IP configuration (DHCP)
Manual configuration
IP Address 192.168.5.200 / Prefix (Netmask) 255.255.255.0

ตัวอย่าง

Enable IPv6 support IP6 ไม่ต้องเลือก
Automatic neighbor discovery
Dynamic IP configuration (DHCPv6)
Manual configuration
IP Address / Prefix

Cancel OK

Network Devi
Active on Boo
Hostname
Set the hostnai
automatica
manually
Miscellaneous
Gateway:
Primary DNS:
Secondary DNS

Release Notes Back Next

eth0 หมายถึง การ์ดแลนใบที่ 1 ใช้เชื่อมต่อกับ CONSUMER BOX ของดาวเทียม iPSTAR ให้คลิกเลือก eth0 แล้วคลิกที่ปุ่ม Edit ให้คลิกเครื่องหมาย เลือกเอา IP4 ส่วน IP6 ไม่ต้องเลือกให้เอาเครื่องหมายออก เสร็จแล้วให้กรอกหมายเลข IP Address ที่ใช้เล่นเน็ตลงไป เช่นของผมกรอก 192.168.5.200 prefix(NetMask) 255.255.255.0 แล้วคลิกปุ่ม OK

Hostname
Set the hostname:
automatically via DHCP
manually marnda.no-ip.org (e.g., host.domain.com)

Miscellaneous Settings
Gateway: 192.168.5.100
Primary DNS: 203.192.33.34
Secondary DNS:

ตัวอย่าง

Release Notes Back Next

ในส่วนของ Hostname ดังภาพข้างบน ให้เลือกที่ manually พิมพ์ชื่ออะไรก็ได้ มันคือโดเมนเนม เช่น marnda.no-ip.org

GateWay ให้ใช้ของดาวเทียม ของผมคือ 192.168.5.100 DNS ให้ใช้ของดาวเทียม ของผมคือ 203.192.33.34 เมื่อกรอกเสร็จให้ไปกรอกของ eth1 ด้วย

eth1 หมายถึง การ์ดแลนใบที่ 2 เป็นการ์ดแลนวงในเครือข่ายเอาไว้แจกเบอร์ให้กับคอมพิวเตอร์ในโรงเรียนให้คลิกที่ eth1 แล้วคลิกที่ปุ่ม edit กำหนดค่า IP4 ให้กำหนดเป็น 192.168.0.1 Prefix (Netmask) 255.255.255.0 IP6 ไม่ต้องใส่ให้เอาเครื่องหมายออก เสร็จแล้วคลิกปุ่ม ok และคลิกปุ่ม Next

10. เลือกโซนเวลาโดยการคลิกเลือกที่รูปแผนที่ประเทศไทย



คลิกเลือกที่แผนที่ประเทศไทย โซนเวลา BangKok

11. กำหนดรหัสให้ root

CentOS 5

The root account is used for administering the system. Enter a password for the root user.

Root Password: ••••••••••

Confirm: ••••••••••

Release Notes | Back | Next

กรอกรหัสเป็นตัวเลขเหมือนกัน 2 ช่องแล้วคลิกปุ่ม Next

12. เลือก Package Server หรือโปรแกรมที่จะติดตั้งให้กับ Server

CentOS 5

The default installation of CentOS includes a set of software applicable for general internet usage. What additional tasks would you like your system to include support for?

- [ ] Desktop - Gnome
- [ ] Desktop - KDE
- [x] Server

Please select any additional repositories that you want to use for software installation.

- [ ] Packages from CentOS Extras

Add additional software repositories

You can further customize the software selection now, or after install via the software management application.

○ Customize later ◉ Customize now

Release Notes | Back | Next

ให้เลือกดังภาพด้านบนแล้วคลิกปุ่ม Next

13. เลือก โปรแกรมให้เครื่อง server แนะนำเลือกให้หมด



ฝั่งซ้ายให้เลือก Servers ฝั่งขวาให้เลือกโปรแกรม ทั้งหมดเมื่อเลือกแล้วให้สังเกตุดูว่าแต่ละโปรแกรมครบหรือไม่ เช่นผมเลือก FTP Server เมื่อสังเกตุดูเป็นดังนี้ 1 of 3 optional package selectd แสดงว่าถูกเลือกแค่ 1 โปรแกรมให้ทำการคลิกเพิ่มที่ปุ่ม Optional packages ดังหมายเลข 3 เมื่อเลือกหมดทุก โปรแกรมแล้วฝั่งซ้ายมือเลื่อนลงมาที่ Lanquages ฝั่งขวาเลือก thai เพื่อให้ server รองรับภาษาไทย





ระบบกำลังจัดเตรียมพื้นที่

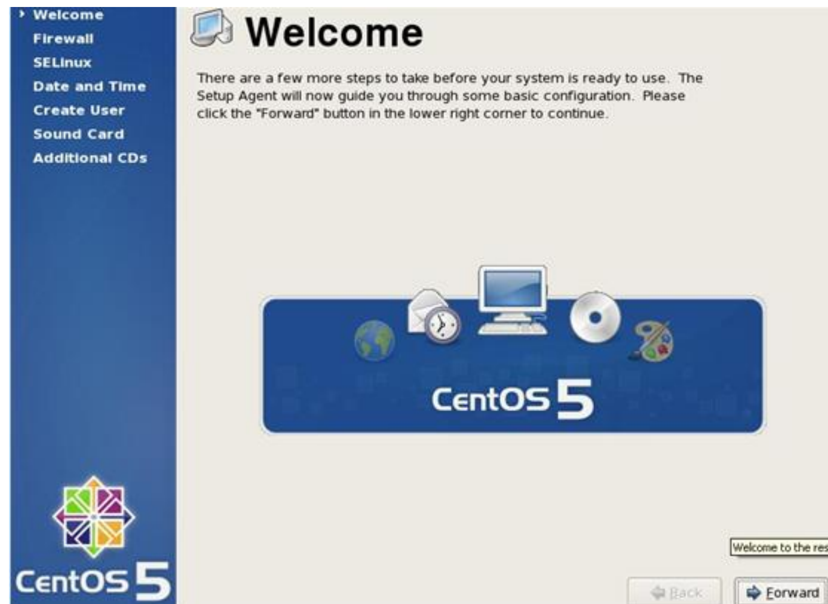
เสร็จแล้วคลิกปุ่ม Next โปรแกรมก็จะเริ่มตรวจสอบพื้นที่พาติชั่น และเตือนให้เราเตรียมแผ่นโปรแกรมทั้งหมด 6 แผ่น และเมื่อเสร็จแต่ละแผ่น CD-ROM จะเลือนแผ่นออกท่เอง ให้นำแผ่นต่อไปใส่ แล้วกด Enter รอให้เสร็จทำไปเรื่อย ๆ จนครบ 6 แผ่น

14. เสร็จสิ้นการติดตั้ง

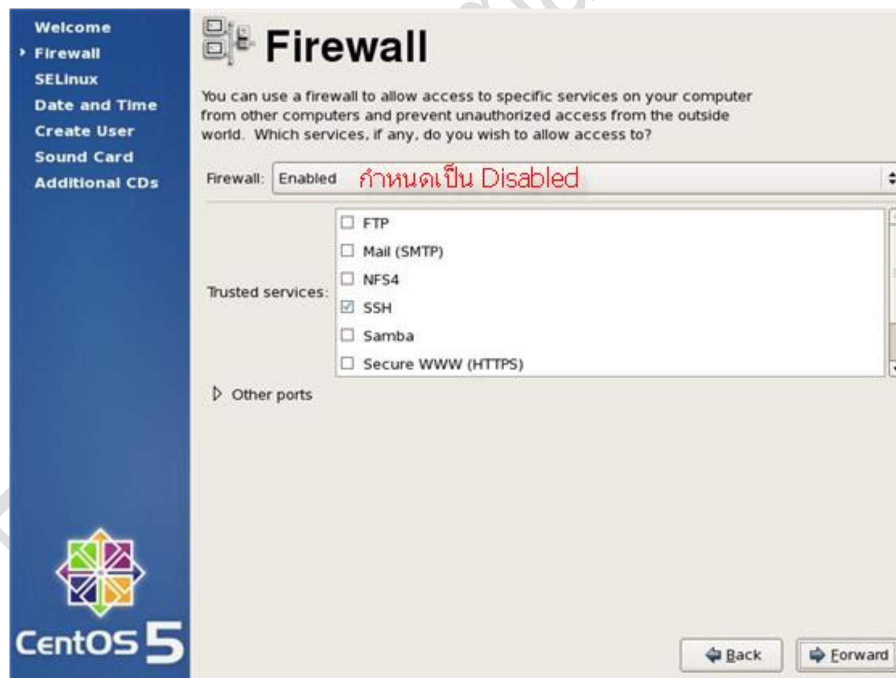


เมื่อเสร็จสิ้นให้คลิกที่ปุ่ม Reboot

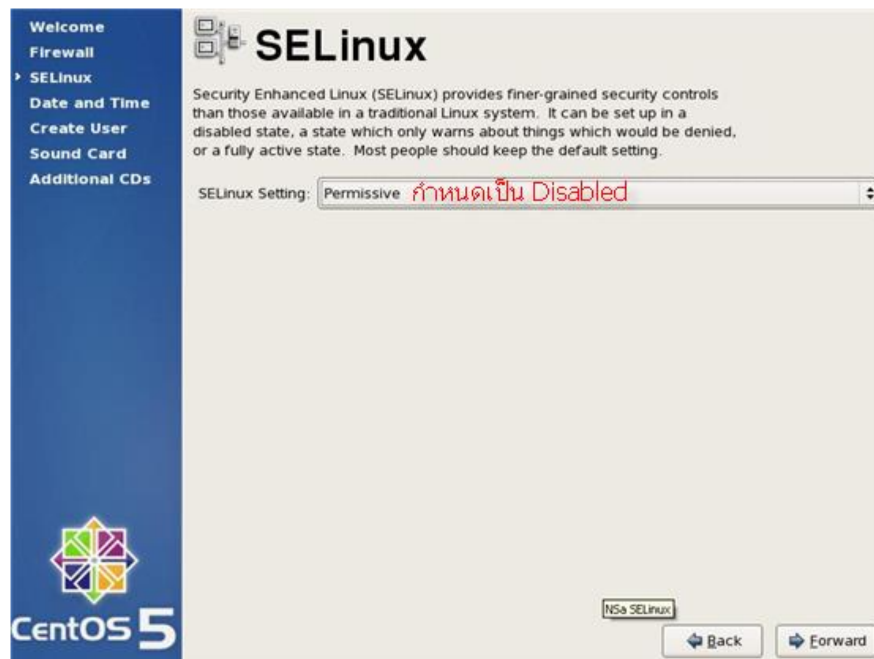
เมื่อเราเข้ามาจะเป็นการกำหนดค่าต่าง ๆ ของ Server โดยที่ Firewall ให้เลือกเป็น Disabled และ SELinux ให้เลือกเป็น Disabled



เข้าสู่การตั้งค่าต่างๆ ของ Server ครั้งแรก



การกำหนดค่า Firewall



การกำหนดค่า SElinux

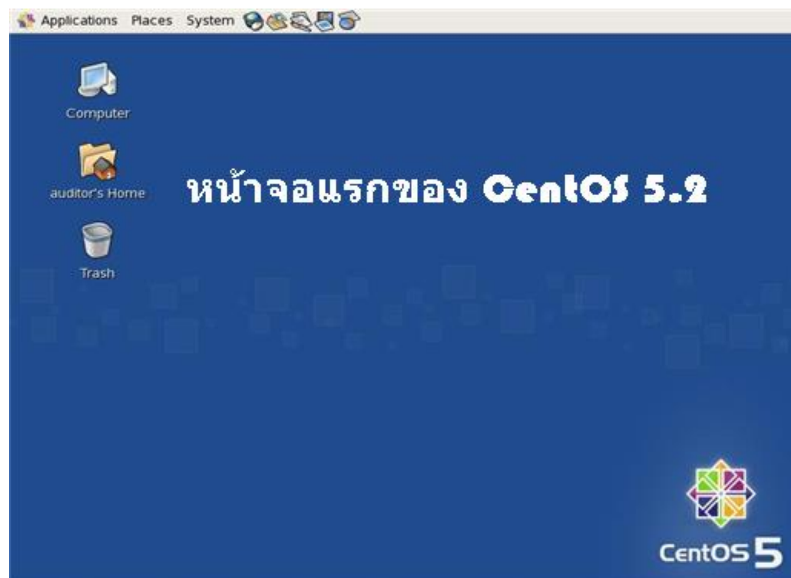
15. เข้าระบบครั้งแรก



Username : root

Password : รหัสที่เป็นตัวเลขที่ท่านกรอกตอนติดตั้ง

16. หน้าจอแรกของระบบ


Applications Places System
Computer
auditor's Home
Trash
หน้าจอแรกของ CentOS 5.2
CentOS 5